H. LILEN

PRATIQUE du micro-ordinateur ADAM

Traitement de texte, programmation BASIC

S. E. C. F.



ÉDITIONS RADIO


PRATIQUE
du
micro-ordinateur
ADAM

H. LILEN

PRATIQUE du micro-ordinateur ADAM

S. E. C. F.



Éditions Radio
9, RUE JACOB - 75006 PARIS
TÉL. 329.63.70

	Imprimerie Berger-Levrault, Nancy Dépôt légal : avril 1984 Éditeur n° 966 - Imprimeur : 779205 I.S.B.N. 2 7091 0944 1

INTRODUCTION

Le micro-ordinateur Adam, commercialisé par la société CBS Electronic (Idéal Loisirs), se distingue manifestement dans la foule des micro-ordinateurs actuellement proposés au public. Il réussit, ce qui n'était guère évident, à innover dans le genre. En effet, il est livré sous forme de trois unités principales qui sont : l'unité centrale, un clavier détachable et surtout, une imprimante de qualité. Le tout, approximativement pour le prix d'une imprimante courante « à marguerite ». Aussi est-ce avec raison qu'on a pu le présenter comme un ordinateur « du 3e type », destiné tout à la fois aux applications professionnelles, domestiques, et aux jeux. Ce qui est exact, vous allez en juger.

Avec ce livre, nous allons vous proposer, en effet, de faire la connaissance d'Adam et d'apprendre à le mettre à votre service, à en exploiter la puissance et les possibilités. Nous supposons au préalable que vous êtes totalement profane en informatique, que vous n'avez jamais manipulé un micro-ordinateur, que vous ignorez ce qu'est un progiciel, la programmation, le traitement de texte ou le Basic, et que peut-être même, la simple évocation de ces termes vous met dans l'anxiété.

Quittez toute appréhension et laissez-vous guider par les pages qui suivent. Nous les avons voulues aussi simples et pédagogiques que possible. En les lisant, vous constaterez vite que tout cela est bien plus facile que vous ne le pensiez. Vous allez aussitôt vous familiariser avec l'ordinateur, l'apprivoiser, et lui faire faire tout ce qui vous passera par la tête. Peut-être commencerez-vous par rédiger, corriger des textes, puis les imprimer (c'est ce qu'on appelle en termes techniques le « traitement de texte »), mais l'ordinateur vous servira aussi à apprendre et à jouer, ou encore à gérer toutes sortes de choses, comme tout ordinateur digne de ce nom.

Vous faut-il disposer d'un micro-ordinateur pour aborder les chapitres qui suivent ? Cela vaut mieux, c'est certain, car vous pourrez alors directement exécuter sur machine nos exercices. Mais ce n'est pas absolument indispensable *en une première lecture, celle qui vous permettra de vous tester vous-même, de répondre à la question « en suis-je capable ? ». Nous osons affirmer que oui. Nous vous prédisons même qu'en lisant ce livre et ici ou là, vous allez partir sur vos propres idées, modifier nos programmes par exemple pour les améliorer, en inventer d'autres... ce à quoi nous vous encourageons d'ailleurs vivement.*

Vous prendrez vite conscience des énormes possibilités que vous offrent les micro-ordinateurs pour toutes vos activités courantes. Ces machines constituent déjà, qu'on le veuille ou non, un point de passage obligatoire. En outre, la micro-informatique fait aussi partie de la culture, ce qui est tout dire.
Bonne lecture, donc, et plein succès.

CHAPITRE I

POUR FAIRE LA CONNAISSANCE D'ADAM

Ce premier chapitre est destiné à vous présenter le micro-ordinateur Adam, détailler sa composition, vous montrer comment on le met en service. Vous apprendrez aussitôt à vous en servir comme d'une (excellente) machine à écrire électronique.

Principaux thèmes étudiés

La micro-informatique fait souvent appel à des termes spécifiques qui découragent le commun des mortels. Une bonne part de ce livre va consister à vous les présenter, c'est-à-dire, en fait, à démystifier l'informatique. Vous constaterez que tout cela, c'est très simple. Mais n'apprenez rien par cœur, c'est inutile : afin de faciliter vos recherches ultérieures, nous avons distingué les mots de ce vocabulaire nouveau, ou ses expressions et idées, sous forme de mots clés. Ils sont répétés en marge du texte qui les évoque. Au début de chaque chapitre, nous vous en dresserons la liste dans l'ordre de leur entrée en scène. A la fin du livre, ils sont tous regroupés par ordre alphabétique avec renvoi à leur page d'appel. Voici leur liste, pour ce premier chapitre. Ne vous laissez pas impressionner par sa longueur apparente !

Branchements
Péritel
Mise en service
Mode « machine à écrire »
Clavier
AZERTY
QWERTY
RETURN ↲
SHIFT
RESET
Cartouches
Menu
Cassettes
Unité à cassettes
Microprocesseur
Mémoires
Mémoires internes
Mémoires centrales
Mémoires externes
Mémoires périphériques
Mémoires vives
RAM
Mémoires mortes
ROM
Chargement d'un programme

1. Adam et ses branchements

Supposons que vous veniez d'acquérir Adam (toutes nos félicitations !). Déballez-le soigneusement de son carton et installez ses divers modules sur une table. Il va vous falloir les interconnecter et relier l'ordinateur à un téléviseur. Deux cas se présentent :

1) Vous disposez du système Adam complet.

2) Vous possédiez déjà la console de jeux ColecoVision et vous avez acquis l'ensemble complémentaire qui, associé à cette console, compose le même système Adam que ci-dessus.

BRANCHEMENTS

Les branchements se différencient légèrement selon le cas mais restent néanmoins très simples. On admirera, en passant, que le fabricant n'ait pas lésiné sur la longueur des câbles.
Notez que les composantes essentielles d'Adam sont :

• ce que l'on appelle *l'unité centrale :* c'est le bloc contenant l'ordinateur proprement dit,

• *l'imprimante,* plus volumineuse que l'unité centrale,

• *le clavier,*

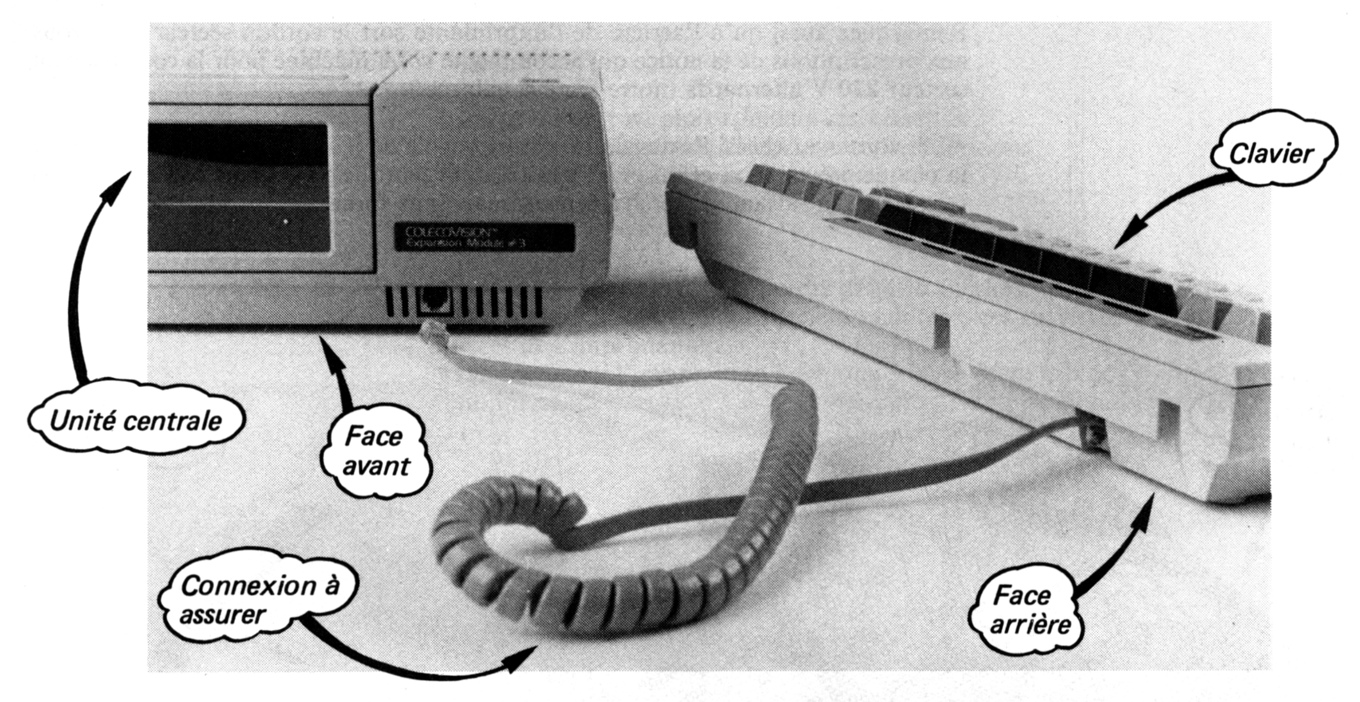
• *les deux petits claviers numériques* servant également de manettes de jeux.



Vue d'ensemble du système Adam, ici en version complémentaire à la console de jeux. Celle-ci apparaît à l'arrière de l'unité centrale. Dans le cas d'un système Adam complet, il n'est plus besoin de la console de jeux mais par contre, l'unité centrale est plus volumineuse.

Les connexions à assurer sont les suivantes :

— Il vous faut relier le clavier à l'unité centrale à l'aide d'un câble extensible.



Enfichez simplement le câble extensible d'un côté, à l'arrière du clavier, de l'autre, sur l'avant de l'unité centrale.

— Connectez ensuite l'unité centrale à l'imprimante : enfichez simplement l'extrémité du câble qui sort de l'imprimante et qui se termine par un connecteur allongé à neuf broches femelles, en deux rangées, dans le connecteur mâle correspondant et situé sur le côté de l'unité centrale.

Imprimante vue de derrière

Pour alimenter votre système

Unité centrale, vue de l'avant

Connexion à assurer

Remarquez ici, l'interrupteur général d'arrêt-marche

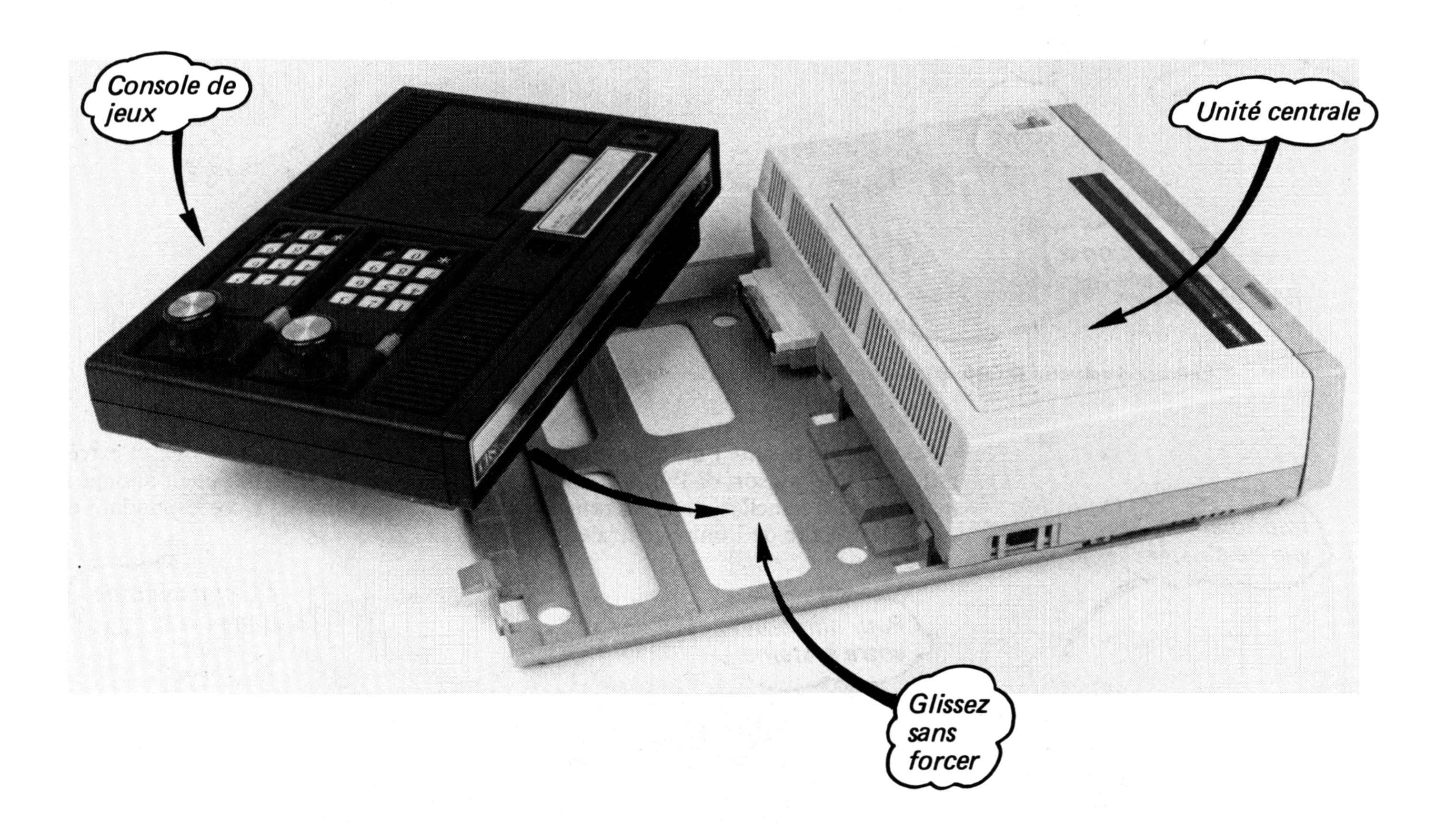
Secteur 220 V

Transformateur éventuel 220/110 volts

Connexion de l'imprimante à l'unité centrale et au secteur.

Remarquez aussi qu'à l'arrière de l'imprimante sort le cordon secteur. Fiez-vous aux prescriptions de la notice qui accompagne *votre* machine pour la connexion au secteur 220 V alternatifs (notre secteur habituel).

— Si vous avez choisi l'extension à votre console de jeux, placez cette dernière sur la plaque prévue à cet effet, et enfichez l'unité centrale dans la prise interface de la console en la faisant glisser fermement mais sans forcer.



Comment relier la console de jeux à l'extension, dans le cas où vous auriez choisi cette formule.

— Branchez les manettes de jeux, soit sur l'unité centrale, soit sur la console de jeux si tel est le cas.

PERITEL

— Enfin, reliez l'ordinateur au téléviseur, via sa prise « Péritel ». Le mot Péritel provient de *péri-télévision ;* dans leur grande sagesse, nos législateurs ont imposé sur tous les téléviseurs français, et ce depuis 1981, la présence d'une telle prise, que vous trouverez généralement située à l'arrière de l'appareil. Elle sert aussi à connecter un magnétoscope ; c'est une prise allongée, à deux rangées de connexions, femelle sur le téléviseur. Son avantage essentiel consiste à procurer des images d'une excellente qualité, bien supérieure à ce que vous obtiendriez en passant par l'antenne.

Saisissez-vous donc du câble *ad hoc* que vous connecterez, d'une part à l'unité centrale (pour un système Adam complet) ou à la console de jeux (dans le cas de l'extension), et d'autre part au téléviseur.
Ce sera tout pour les connexions. Notez aussi qu'avec Adam vous ont été fournies trois cassettes digitales.

Attention : ne placez *jamais* ces cassettes sur l'unité centrale, l'imprimante, le téléviseur ou tout dispositif créant un champ magnétique.

Vous risqueriez de leur faire perdre des informations et de rendre leurs programmes inutilisables. Il faut impérativement maintenir ces cassettes à plus de trente centimètres de l'imprimante..
Examinez aussi l'imprimante ; débloquez-là en ôtant le bloc de polystyrène expansé qui l'immobilise, et si ce n'est déjà fait, mettez en place le ruban et la tête d'impression.

2. Mise en service

Tout est-il parfaitement connecté ? Installez-vous confortablement devant le système car on va le mettre en service. Pour cela, branchez-le sur le secteur.

Vérifiez qu'aucune cassette ni cartouche n'est insérée dans le système et mettez votre téléviseur en service *sur n'importe quelle chaîne :* la prise Péritel a priorité. Si toutefois une station trop puissante trouvait le moyen de manifester sa présence, réglez le téléviseur sur un canal libre. Puis mettez le système en service en manœuvrant l'interrupteur général situé à l'arrière de l'imprimante.

MISE EN SERVICE

Si vous disposez de l'extension, connectée à la console de jeux, mettez aussi l'interrupteur de la console en position « marche ».

Vous pourriez, d'ailleurs, le laisser ainsi en permanence. Aussitôt, l'imprimante manifeste sa présence : le chariot se déplace et vient en position de départ alors que sur l'écran du téléviseur s'affiche l'image qui vient couronner vos efforts de succès. En outre, le petit voyant rouge, en bas et à la droite du clavier, s'allume.

Notez tout de suite la séquence des opérations :

1) D'abord, mettre le téléviseur en service.

2) Puis l'ordinateur.

A l'extinction, vous procèderez à l'inverse, le téléviseur étant mis à l'arrêt en dernier.

Si vous ne respectiez pas cette procédure, vous risqueriez de provoquer des incidents (que l'auteur n'a jamais constatés et n'a jamais pu créer non plus, en dépit de sa bonne volonté). Mais vous voilà prévenu.

3. Une bonne machine à écrire électronique

La première image qui apparaît à l'écran est la suivante :

Réglette servant à marquer les positions horizontales

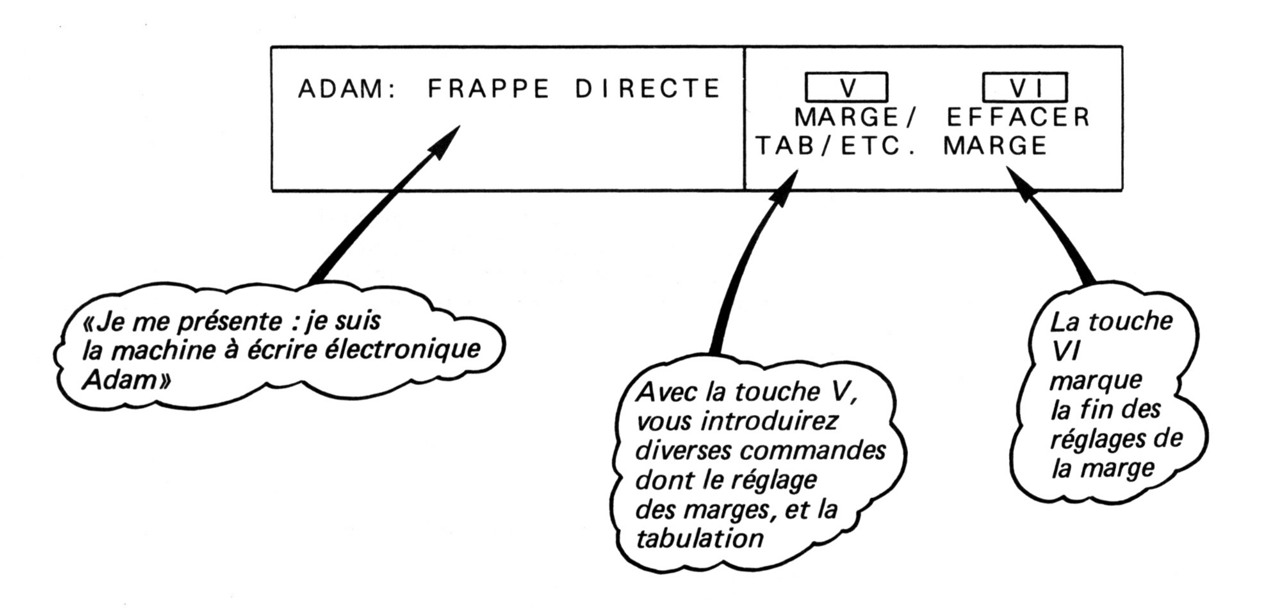
Emplacement pour le texte

Fenêtre noire dans laquelle votre texte va apparaître lors de la frappe

Cette fenêtre simule le «rouleau d'entraînement» de l'imprimante

Message et commandes

Le fond est bleu-vert et diverses couleurs se manifestent déjà. Les lignes réservées aux messages fournissent des informations que nous vous traduisons librement :



MODE " MACHINE A ECRIRE »

Insérez donc une feuille de papier dans l'imprimante (fallait-il réellement vous le dire ?), et frappez ce que vous voulez sur le clavier (il fonctionne comme celui d'une machine à écrire courante) ; par exemple, *bonjour*. Aussitôt, ce texte apparaît en blanc sur fond noir dans la zone de l'écran réservée au texte frappé, juste au-dessus de la zone des messages, et est frappé en même temps par l'imprimante. Vous voici donc en possession de l'équivalent d'une machine à écrire électronique (notez que ce n'est pas encore cela, le traitement de texte).

Profitons-en pour jeter un coup d'œil sur le clavier. Ce clavier de 75 touches est divisé en plusieurs sections :

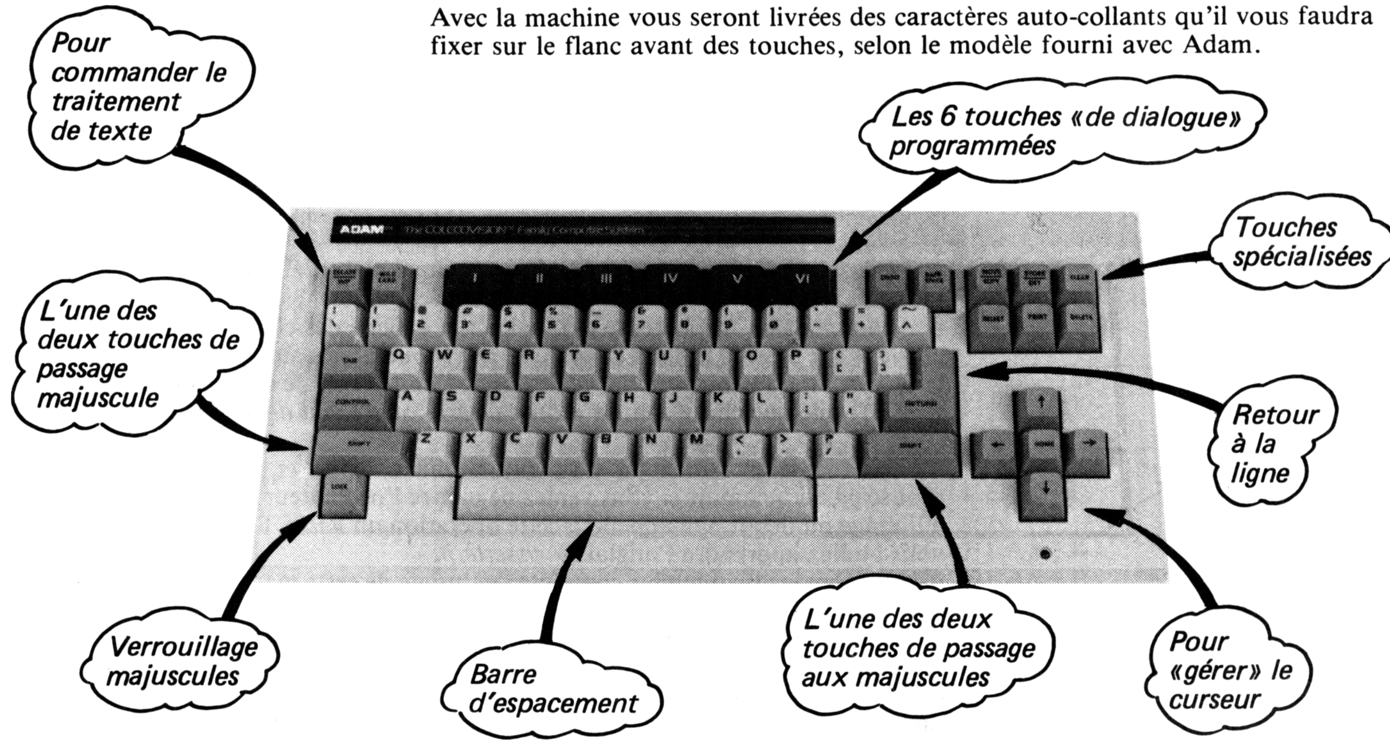
CLAVIER

• Un clavier alphanumérique courant, avec des touches claires. Il ressemble à celui de toute machine à écrire.

• Une rangée de touches supérieures noires, marquées en romain de I à VI. Il s'agit de touches de fonctions encore appelées *touches dialogue* et dont on va voir l'usage.

• Des touches en gris foncé servant à des fins diverses qu'on va étudier.

Avec la machine vous seront livrées des caractères auto-collants qu'il vous faudra fixer sur le flanc avant des touches, selon le modèle fourni avec Adam.



Gros plan sur le clavier. Attention : l'auteur ne disposait que d'un clavier QWERTY alors que vous aurez probablement acquis un AZERTY/QWERTY. Négligez donc la distribution des touches alphanumériques ; pour cela, reportez-vous à *votre* clavier. Par contre, attachez-vous aux autres touches, pour fonctions diverses. Nous étudierons leur rôle au fur et à mesure des besoins.

Arrêtons-nous un instant sur la section alphanumérique. La première rangée supérieure *alphabétique* comportera, sur les faces avant des touches :

AZERTY

QWERTY

• Les lettres auto-collantes A, Z, E, R, T, Y, etc. C'est là le clavier aux normes françaises, appelé sans imagination clavier AZERTY (prononcez « *azerti* »). C'est le même que sur les machines à écrire.
• Sur le dessus des touches, vous lirez Q, W, E, R, T, Y, etc. C'est le clavier aux normes américaines et dans ce cas, l'emplacement des caractères est un peu différent et il n'y a pas d'accents (ou, comme l'on dit, de *caractères accentués).* C'est la version dite QWERTY (prononcez « *cuverti* »*).*

Adam, grâce à ses auto-collants, disposera donc d'un clavier double AZERTY/QWERTY.Malheureusement, l'auteur a dû rédiger ce livre *avant* sa commercialisation, ce qui explique que les photographies d'accompagnement montrent un clavier QWERTY uniquement. De toutes façons, notre but n'est pas de vous enseigner le noble art de la dactylographie, aussi va-t-on passer aux autres touches.

SHIFT

Le passage aux majuscules se fait, comme sur les machines à écrire, en frappant l'une des deux touches existantes et symétriques de *passage aux majuscules* et marquées ici SHIFT (en gris foncé, de part et d'autre de la barre d'espacement) ou indiquées par une flèche large vers le haut. Frappez l'une de ces deux touches, au choix, maintenez-là enfoncée, puis frappez une lettre quelconque : elle viendra en majuscules. Le mot SHIFT anglais signifie approximativement « basculement dans l'autre mode » ; parce qu'il s'est internationalisé de fait, on l'emploiera couramment, y compris avec son affreux adjectif franglais « shifté » ; que les puristes nous pardonnent.

Le verrouillage en position majuscules s'opère avec la touche marquée LOCK *(verrouillage,* en anglais). Un coup pour verrouiller et tout vient en majuscules, un coup pour déverrouiller et revenir aux minuscules.
Remarquez que certaines touches (numériques ou de symboles) possèdent un double marquage. Ainsi que sur les machines à écrire, le code du bas apparaît en position *minuscules,* celui du haut en position *majuscules.*

RETURN

Le retour à la ligne est obtenu avec la touche RETURN, qu'il n'est pas nécessaire de traduire et, marquée aussi d'une flèche à angle droit (↵). Mais vous en savez assez sur la clavier pour un début. Exercez-vous à ces diverses possibilités.

Ne vous privez pas de frapper à volonté sur le clavier : vous ne risquez en aucun cas de nuire à votre ordinateur.

Rappelez-vous qu'il ne s'agit plus de mécanique mais d'électronique (seuls les contacts des touches peuvent être qualifiés de mécaniques). Si, toutefois, vous frappiez une combinaison incohérente et que l'écran affiche des choses bizarres, ou que le clavier fasse ensuite le mort (le système ne répond plus à aucune frappe), retournez-vous vers un poussoir situé sur le dessus de l'unité centrale et marqué COMPUTER RESET, quelque chose comme *remise à zéro,* ou *à l'état de départ de l'ordinateur.*

RESET

Manœuvrez-le en l'amenant vers vous un instant puis en le relâchant ; aussitôt, tout se passera comme si vous veniez de mettre l'ordinateur en service, en revenant à l'image du début. On appellera cette opération un RESET, ce qui se prononce, si vous voulez apprendre l'anglais « *rissette* ».
Nous réservons l'étude plus complète de l'usage de cette machine à écrire électronique au chapitre suivant, consacré au traitement de texte, mais avant d'en terminer avec ce thème, voyons l'usage des deux claviers numériques, servant à la fois de manettes de jeux. Tous deux ont ici les mêmes fonctions : si vous frappez sur les chiffres, ce sont des chiffres qui seront frappés et affichés ; l'astérisque vous fournira un point (les Américains utilisent un *point décimal* là où nous, nous plaçons une *virgule décimale :* nul n'est parfait) ; enfin, le symbole dièse (#) provoquera un retour à la ligne, tout comme la touche RETURN ↵.

4. Les cartouches de jeux

Au-dessus de l'unité centrale d'Adam apparaît une trappe servant à enficher une cartouche qui est un module contenant des composants électroniques dans lesquels ont été stockés des programmes, et par exemple des jeux. L'auteur a beaucoup aimé le jeu « Donkey Kong ».

CARTOUCHES

Si vous disposez de l'extension, la trappe pour les cartouches se trouve sur la console de jeux.

Saisissez-vous d'une cartouche, de jeu, par exemple. Mais attention : une précaution d'emploi s'impose.

Avant d'insérer ou d'ôter une cartouche de son réceptacle, il faut couper l'alimentation.

Si vous vous servez de l'extension, il suffit de couper l'alimentation de la seule console de jeux. Coupez donc l'alimentation, enfichez la cartouche dans son réceptacle *son étiquette vous faisant face,* puis rétablissez l'alimentation. L'écran qui apparaît est toujours celui de la machine à écrire électronique aussi, pour passer au jeu, actionnez la commande marquée REMISE EN JEU sur la console, RESET CARTRIDGE sur l'unité centrale, selon votre cas. Aussitôt, l'écran annonce le jeu ; il

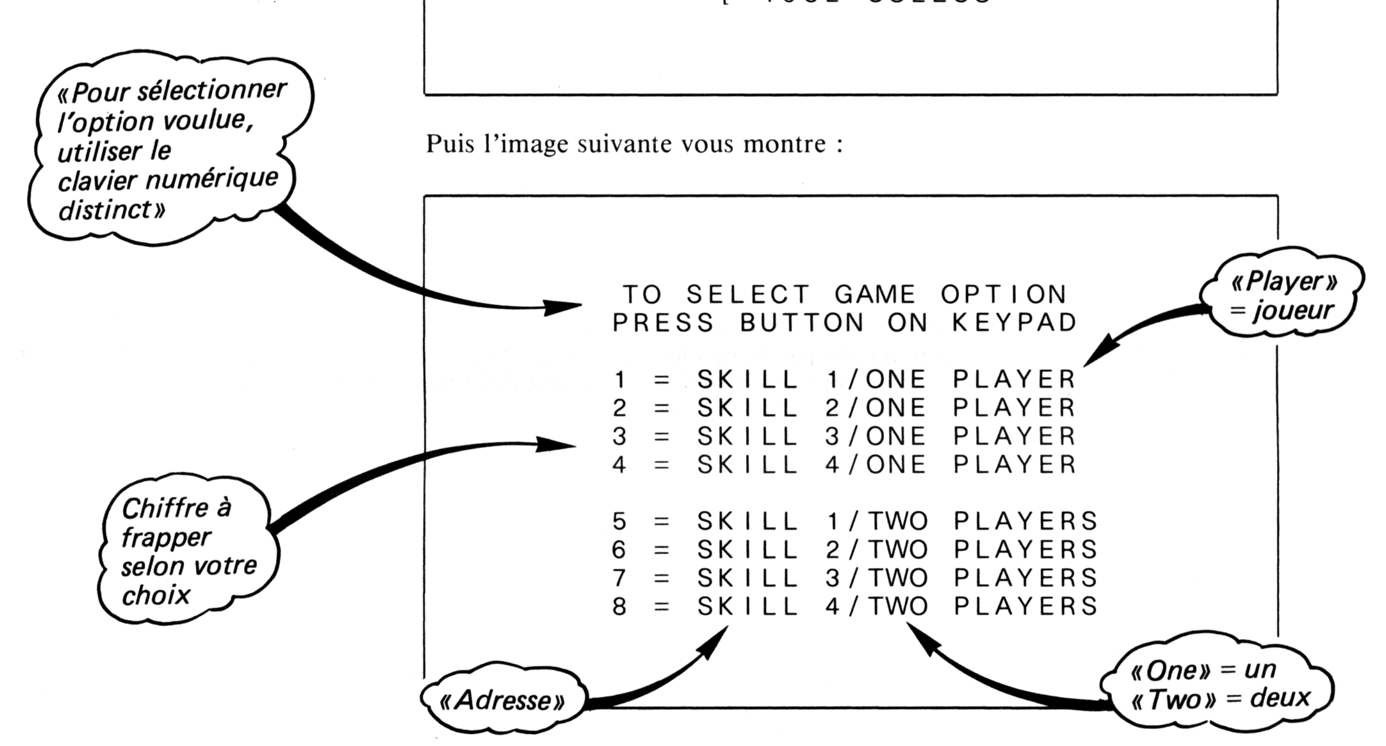
vous faudra en lire les règles et vous borner à suivre les indications qui vous sont fournies (sélection du niveau de difficulté, du nombre de joueurs...) ; par exemple et pour Donkey Kong, l'écran affiche un moment le titre :

```
COLECOVISION

PRESENTS NINTENDO'S
DONKEY KONG

[ 1982 COLECO
```

Puis l'image suivante vous montre :



Dès lors, il ne vous reste plus qu'à frapper le numéro demandé mais en vous servant, cette fois, du clavier numérique-commande de jeux uniquement.

MENU

Un tel choix d'options, affiché à l'écran, s'appelle « un menu ». Comme au restaurant.

Le guide d'emploi du jeu vous indiquera à quoi servent les touches du bloc numérique, les poussoirs et le levier directionnel. Nous vous souhaitons de trouver à jouer autant de plaisir que l'auteur de ce livre. Mais surtout, n'oubliez pas, après avoir joué, de couper l'alimentation *avant* d'ôter la cartouche de jeux, faute de quoi celle-ci risquerait d'être endommagée.

5. Les cassettes de jeux

CASSETTES

Vous pouvez également jouer avec les « super-jeux » d'Adam, sur *bande magnétique,* en cassettes. Il s'agit de cassettes semblables à celles que vous employez avec un magnétophone mais elles sont, là, de très haute qualité et dites « digitales ». Commençons par jouer car, on le verra, ces cassettes peuvent contenir des programmes tout ce qu'il y a de sérieux, tels que des programmes de gestion, ou d'enseignement. Vous remarquerez que ces sujer-jeux possèdent une définition graphique étonnante ; certains sont même en « trois dimensions » et offrent de très

nombreuses phases de jeu avec des niveaux de difficulté variés. Ajoutons que la plupart sont la reproduction (à domicile) des meilleurs « jeux de café ». Adam vous est normalement livré avec trois cassettes :

- l'une marquée *Smart Basic :* elle nous servira pour programmer ;
- la seconde s'appelle *Super Game Pack :* c'est celle qui nous intéresse pour jouer, avec un jeu intitulé Buck Rogers Planet of Zoom ;
- la troisième est une cassette vierge.

UNITE A CASSETTES

Le lecteur-enregistreur de cassettes, qu'on appellera *unité à cassettes,* est situé à gauche, sur le front avant de l'unité centrale (un emplacement libre est réservé à une seconde unité à cassettes). Ouvrez le volet de protection en poussant vers l'arrière sur le bouton marqué EJECT *(éjection)* qui le surmonte. Introduisez la cassette de jeux comme vous le feriez avec un magnétophone puis refermez le volet de protection en le repoussant simplement.

Attention : cette fois, il ne faut jamais mettre l'ordinateur sous tension ou hors tension lorsqu'une cassette est en place.

A la mise en marche ou à la remise à l'arrêt, en effet, il se produit ce qu'on appelle des « transitoires de commutation » qui risqueraient de nuire aux informations contenues par la bande. La cassette étant en place, faites un « RESET COMPUTER », une remise de l'ordinateur à l'état initial en actionnant le poussoir situé *sur le dessus* de l'unité centrale.

Si une cassette est en place lorsque vous faites un RESET, l'ordinateur va commencer par la lire et exécutera ce qu'elle lui prescrit.

Lorsqu'il n'y a pas de cassette, on l'a vu, l'ordinateur part en mode « machine à écrire ». Mais ici, examinez ce qui se passe après le RESET : l'unité à cassettes s'anime, la bande défile, à plusieurs vitesses semble-t-il ; c'est que l'ordinateur procède à la recherche, puis à la lecture d'un programme. Dès lors, il affichera ce qu'il lui aura enjoint de faire, par exemple et dans le cas de l'auteur, le jeu Buck Rogers Planet of Zoom dans notre cas.
Remarquez que la lecture d'un programme de jeu sur cassette est moins rapide qu'avec les cartouches ; dans ce dernier cas, il est instantané alors qu'il vous faudra patienter plusieurs secondes avec les cassettes ; cela est aussi dû, devons-nous préciser, au fait que le jeu est plus complet et que l'ordinateur ne le lit que par sections : si vous réussissez les premières phases, il prépare les suivantes. A nouveau et pour jouer, tout dépendra des règles du jeu ; mais nous vous faisons pleinement confiance sur ce point (et sur d'autres !).
N'oubliez pas, après avoir joué, d'extraire la cassette de son unité et ce, *avant* de mettre l'ordinateur hors tension.

6. Mais comment tout cela fonctionne-t-il ?

Non, nous n'allons pas vous infliger un cours d'informatique ou d'électronique ; nous voulons simplement vous présenter quelques mots spécifiques dont l'usage revient souvent, ne serait-ce que sur les documents publicitaires. Sachez toutefois que vous pouvez sans inconvénient majeur vous dispenser de lire ce qui suit.

MICROPROCESSEUR

Sachez, tout d'abord, que l'âme de tout micro-ordinateur est un minuscule circuit électronique qui tient dans le creux de la main et s'appelle un « microprocesseur ». Ajoutons que c'est un composant fabuleux. Celui qui équipe Adam porte le nom de « Z80 » ; il travaille caractère par caractère, un caractère correspondant, par exemple, à la frappe d'une touche au clavier. On dit qu'il est à mots de 1 octet, ou de 8 bits ce qui est synonyme.

MEMOIRES

Mais pour que ce microprocesseur exécute vos ordres, il doit s'appuyer sur des circuits annexes qui lui serviront de *mémoires.* Celles-ci stockent des programmes de jeux, de traitement de texte, et toutes sortes d'informations. Là encore, il s'agit de minuscules circuits électroniques (des « circuits intégrés ») capables de répondre instantanément aux sollicitations du microprocesseur.

MEMOIRES INTERNES CENTRALES EXTERNES PERIPHERIQUES

Ces circuits mémoires sont appelés des mémoires « internes » à l'ordinateur, ou encore ses « mémoires centrales ».
Par opposition, les mémoires à bande magnétique (cassettes) sont appelées des mémoires « externes » ou encore « périphériques ».

MEMOIRES VIVES RAM MORTES ROM

Ces mémoires centrales sont classées en deux familles essentielles, pour ce qui nous concerne :

• *Les mémoires vives* dans lesquelles vous pouvez enregistrer des informations à volonté, les lire, les modifier. Elles sont affligées d'un sigle d'origine américaine qui est RAM (prononcez comme « *rame* » de train).

• *Les mémoires mortes* qui ont, elles, été enregistrées une fois pour toutes par le fabricant. Vous ne pouvez plus modifier leur contenu, vous ne pourrez que les lire ; elles se comportent comme les pages d'un livre, vu sous cet aspect. Leur nom américain est ROM (prononcez comme dans « Oh *Rome,* unique objet... », etc.).

C'est dans des ROM que *CBS Electronic* a logé le programme *indélébile* de traitement de texte. Les cartouches de jeux contiennent aussi des mémoires mortes qui vont s'ajouter à celle de la mémoire centrale ; c'est pourquoi leur lecture est si rapide.
Ces ROM sont bien utiles, comparées aux RAM : les mémoires vives, qui peuvent voir leur contenu modifié, ont la fâcheuse habitude de perdre toutes leurs informations dès qu'on coupe l'alimentation de l'ordinateur. On dit qu'elles sont *volatiles.* Tel n'est pas le cas des ROM, qui ne sont pas volatiles. Les bandes magnétiques ne sont pas volatiles, elles non plus.

Sachez encore que pour que l'ordinateur puisse exécuter un programme, il faut *impérativement* que celui-ci soit stocké dans ses mémoires centrales, « mortes » ou « vives ». S'il se trouve sur bande magnétique, l'ordinateur devra au préalable le lire et en transférer une copie dans ses mémoires centrales *vives ;* c'est ce qu'il a fait avec la cassette de jeu. Ce n'est qu'alors qu'il pourra l'exécuter.

CHARGEMENT D'UN PROGRAMME

Cette opération de lecture d'un programme sur cassette et de transfert en mémoire centrale s'appelle « le chargement ». On dit qu'on « charge un programme ».

Combien de mémoire votre Adam met-il à votre disposition pour charger vos programmes ? Une « cellule » de mémoire peut accueillir un seul caractère frappé au clavier ; c'est ce qu'on appelle *un octet* (ou « 8 bits ») ; or, Adam dispose de plus de 80 000 cellules de mémoires vives. Exactement, il est équipé de 80 K octets de RAM ; sachant qu'en informatique, le coefficient K vaut 1024, cela représentera exactement $1024 \times 80 = 81920$ cellules, ou octets. Bravo de nous avoir suivis jusque-là !

Caractéristiques résumées
du micro-ordinateur Adam

- Microprocesseur Z80 travaillant sur 8 bits.
- Mémoire centrale vive de 80 K extensible à 144 K.
- Clavier de 75 touches, dont 6 touches de dialogue, 5 touches de gestion du curseur, 8 touches programmées pour traitement de texte. — Deux claviers numériques/commandes de jeux distincts.
- Affichage sur l'écran de 24 lignes de 36 caractères, en traitement de texte, de 31 caractères en « mode texte ». — Graphique sur 40 × 40 et, en haute résolution, de 256 × 192 points. — 16 couleurs et 32 « lutins » (« sprites »).
- 3 voies son sur 5 octaves et générateur de bruit.
- Imprimante à marguerite interchangeable, 10 caractères par seconde, sur papier feuille à feuille ou accordéon. — Imprime jusqu'à 80 caractères par ligne.
- Interfaces pour cartouches de jeux, bus d'extension.
- Une unité à cassettes magnétiques incorporée. — Cassettes numériques, à déroulement rapide, stockant 500 K octets.
- Traitement de texte en ROM (32 K). — Basic compatible Applesoft (de Apple) sur cassette. — Module d'extension CP/M en option.
- En option également, une seconde unité à cassettes, un modem, une unité à disquettes de 5 1/4 pouces (avec compatibilité *IBM,* PC-junior).

Questions sur le chapitre I

Peut-être espériez-vous en avoir terminé avec le chapitre I ? Que non ! Nous allons maintenant vous proposer quelques questions afin de vérifier si vous avez bien assimilé ce qui était essentiel dans ce chapitre. Vous allez le constater, c'est simple mais toujours très important. Bornez-vous à répondre aux questions suivantes en cochant la case correspondant à la bonne réponse.

1. L'unité centrale, c'est :
   ☐ Le bloc comprenant l'ordinateur proprement dit
   ☐ L'imprimante

2. Le clavier AZERTY est le clavier :
   ☐ Qui a été inventé par Don Azerty
   ☐ Correspondant aux normes françaises

3. Pour basculer en majuscules, il faut frapper sur la touche :
   ☐ RETURN, puis frapper sur le caractère qui doit venir en majuscule
   ☐ SHIFT, la laisser enfoncée, puis frapper sur le caractère qui doit être en majuscule

4. La commande RESET (COMPUTER) provoque :
   ☐ Une remise à l'état initial du système
   ☐ Le lancement d'un jeu sur cartouche

5. Un menu, c'est un :
   ☐ Choix entre diverses options
   ☐ Un titre de programme

6. Pour insérer ou extraire une cartouche, il faut obligatoirement :
   ☐ Mettre l'ordinateur hors tension au préalable
   ☐ Mettre l'ordinateur sous tension au préalable

7. A-t-on le droit de laisser une cassette dans son unité lorsqu'on met l'ordinateur hors, ou sous tension :
   ☐ Oui
   ☐ Non

Avez-vous répondu à toutes ces questions qui restent fondamentales, rappelons-le ? Si oui, vérifiez vos solutions avec celles que nous vous donnons page 38.

CHAPITRE II

FAITES DU TRAITEMENT DE TEXTE

Avez-vous du courrier à taper ? Des notes ou des rapports à rédiger ? Peut-être êtes-vous homme (ou femme) de lettre, romancier, poète ? Ou alors gestionnaire, responsable d'entreprise, membre éminent d'une profession libérale, commerçant ou artisan ? Dans tous les cas, vous avez sûrement à rédiger. Il en ira de même si vous êtes enseignant ou élève. Pourquoi ne pas directement composer un devoir ou une thèse sur votre Adam ? Ce chapitre va vous montrer comment, avec une facilité surprenante. le « traitement de texte » vous permet de créer un document, le corriger, le modifier, bref, faire un tas de choses qui se révèlent complexes, longues ou fastidieuses lorsqu'il faut les exécuter à la main. En plus, l'ordinateur conservera tout ce que vous ferez dans un ordre impeccable.

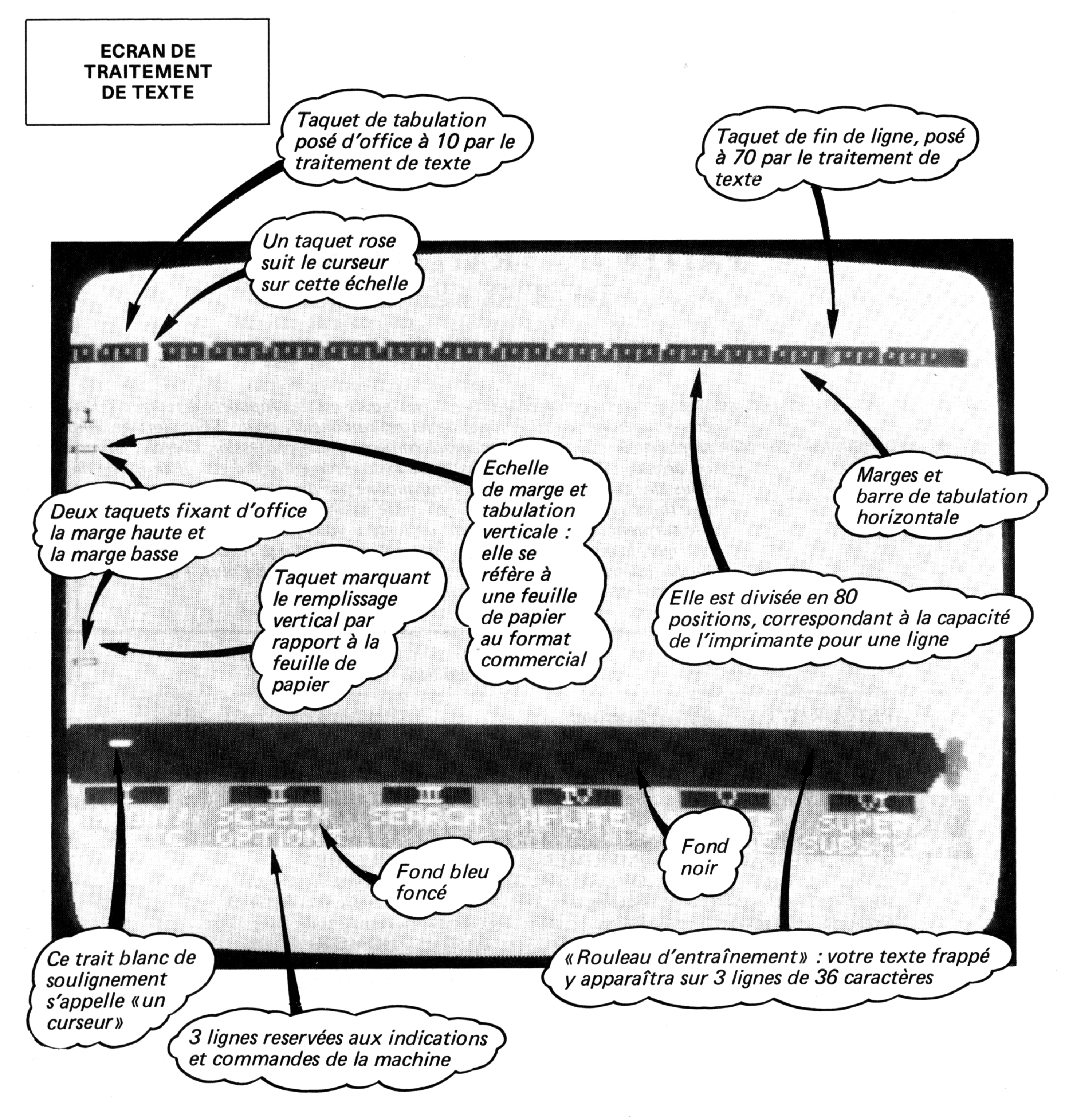
Principaux thèmes étudiés

RETOUR/T/T	Insertion	Restitution
Ecran de traitement de texte	INSERER	Lecture
Curseur	Suppression	Chargement
Rouleau d'entraînement figuré	SUPPRESS	ACCES
Effacement d'un caractère	Soulignement en rouge	FICHIER
RETOUR/ESPACE	SOULIGNER	EFFACER
Retour à la ligne	ESCAPE	Contre-ordre
RETURN (↲)	IMPRIMER	ERREUR
Création d'un texte	ORDRE FINAL	Marges
Modifications ou corrections	Sauvegarde	Chiffre 0 et lettre O
Gestion du curseur	Rangement	Tabulations
	ENREGIS	HOME R
	Fichier	RECHERCHE
	FICHIER	CHANGER

1. Introduction au traitement de texte

RETOUR/ T/T

On va commencer par expérimenter timidement le traitement de texte. Mettez Adam en service, de façon à obtenir le premier écran correspondant au mode « machine à écrire électronique ». Puis, entrez en mode « traitement de texte » en frappant sur la touche à l'extrémité supérieure gauche marquée RETOUR/T/T ; les lettres TT proviennent de traitement de texte. Aussitôt, l'écran se modifie (accompagné en cela par un accord musical sonore) :



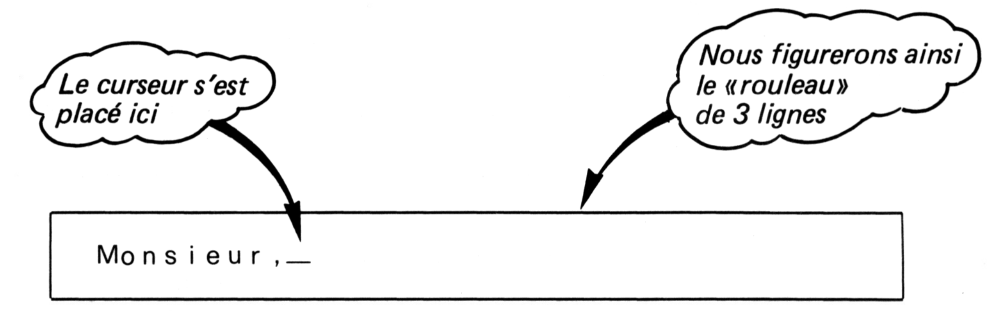
L'écran de traitement de texte, à peu près à l'échelle (les proportions de l'écran sont à peu près de 2/3 pour le rapport hauteur/largeur).

Nous avons noté les points essentiels de cet écran qui doivent retenir votre attention. Sachez qu'à la mise en service, les marges sont préréglées par le traitement de texte, aussi ne va-t-on pas s'en occuper en un premier temps, non plus d'ailleurs que des commandes diverses, qu'on étudiera par la suite. On va plutôt commencer par expérimenter ce que la machine peut faire en frappant un texte quelconque, par exemple une lettre. Commencez par frapper « *Monsieur,* », et observez comment chaque caractère frappé vient s'inscrire dans la zone noire du bas qui simule le « rouleau d'entraînement » de la machine à écrire. A chaque frappe, le *curseur* (ce petit trait blanc de soulignement) se déplace d'un pas vers la droite.

CURSEUR

Le curseur marque toujours l'emplacement où votre prochaine frappe apparaîtra.

Ainsi, ce que vous frappez apparaît sur la ligne supérieure du « rouleau d'entraînement » figuré ; exercez-vous, ce faisant, à employer tous les caractères à votre disposition, majuscules comprises :



ROULEAU D'ENTRAINEMENT FIGURE

Examinez comment le curseur se déplace à chaque frappe, accompagné d'une note musicale.

EFFACEMENT D'UN CARACTERE

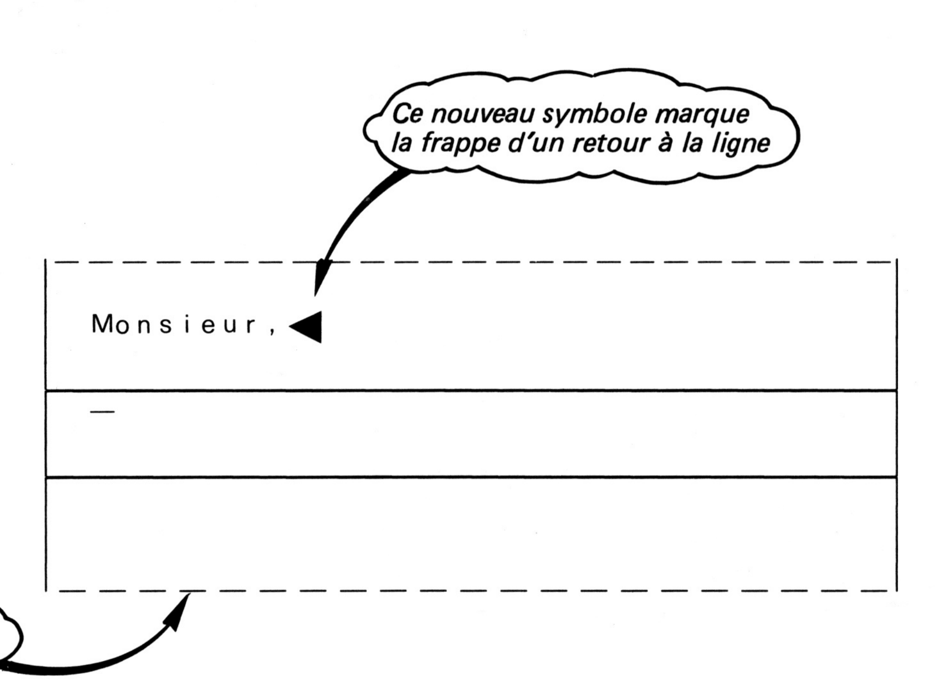
RETOUR/ESPACE

Si vous commettez une erreur de frappe, appuyez sur la touche supérieure marquée RETOUR/ESPACE (« espace arrière »). Le curseur revient d'un pas en arrière et efface le caractère précédent. Refrappez alors le bon caractère.

C'est là la touche d'effacement arrière ; c'est bien plus simple qu'avec tout autre système. Pour aller à la ligne, frappez sur RETURN (↲) ; aussitôt, votre texte est projeté dans l'espace supérieur et le « rouleau d'entraînement » se vide :

RETOUR A LA LIGNE

RETURN (↲)



2. Création d'un texte

Continuez en frappant « *Saviez-vous que le premier micro-ordinateur* » et observez, ici, ce qui va se passer lors de la frappe du mot *micro-ordinateur,* qui déborde de la première ligne *de l'écran :* d'office, le traitement de texte va le propulser sur la seconde ligne *dans son intégralité,* marquant par des points la place qu'il y occupera réellement :

```
Monsieur, ◀
---------------------------------------
Saviez-vous que le premier .........
micro-ordinateur_
```

CREATION D'UN TEXTE

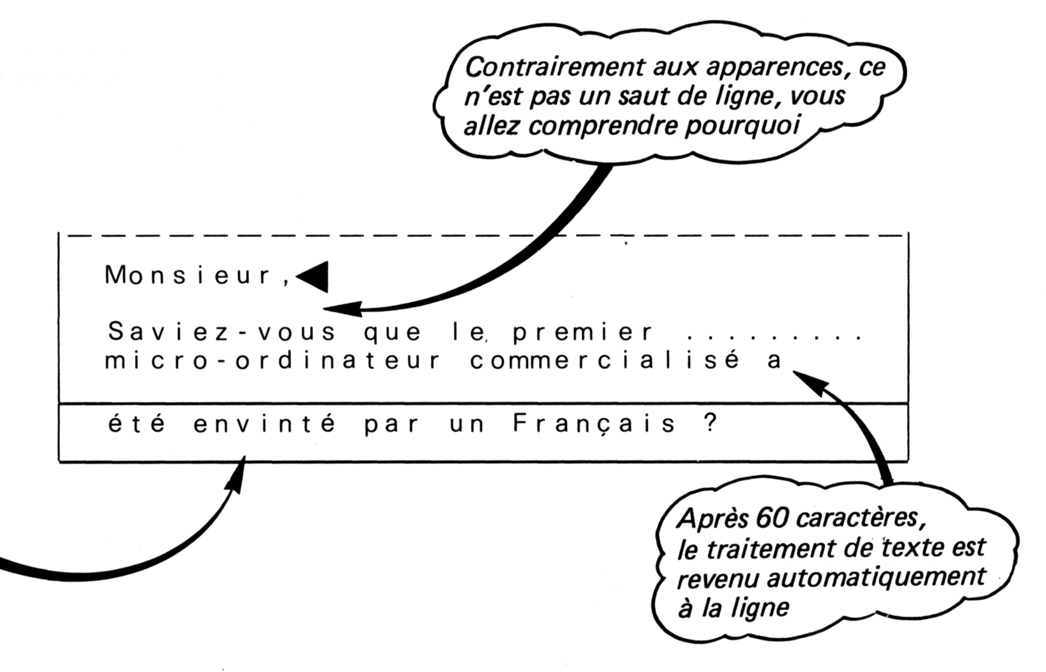
Si vous commettez des erreurs de frappe, corrigez-les en effaçant les caractères précédents ou ignorez-les provisoirement : on verra plus loin comment ce traitement de texte permet leur correction avec la plus grande facilité. Continuez à frapper, toujours en examinant ce qui se passe.

Vous verrez qu'après 60 caractères ponctués par une note musicale, le texte dans le rouleau passe d'office dans la zone texte.

En effet, le traitement de texte a placé d'office des marges en 10 et 70, horizontalement, ce qui signifie que l'impression *sur la feuille* fera apparaître des marges gauche et droite identiques, le texte étant alors de 70 – 10 = 60 caractères par ligne.

Vous n'avez donc même pas à frapper de RETURN : le RETURN ne vous servira qu'à marquer la fin d'une ligne arbitrairement terminée, d'un paragraphe (ou à sauter une ligne.

Frappez donc à la suite le texte que nous continuons à vous proposer (ou encore un texte de votre crû) :

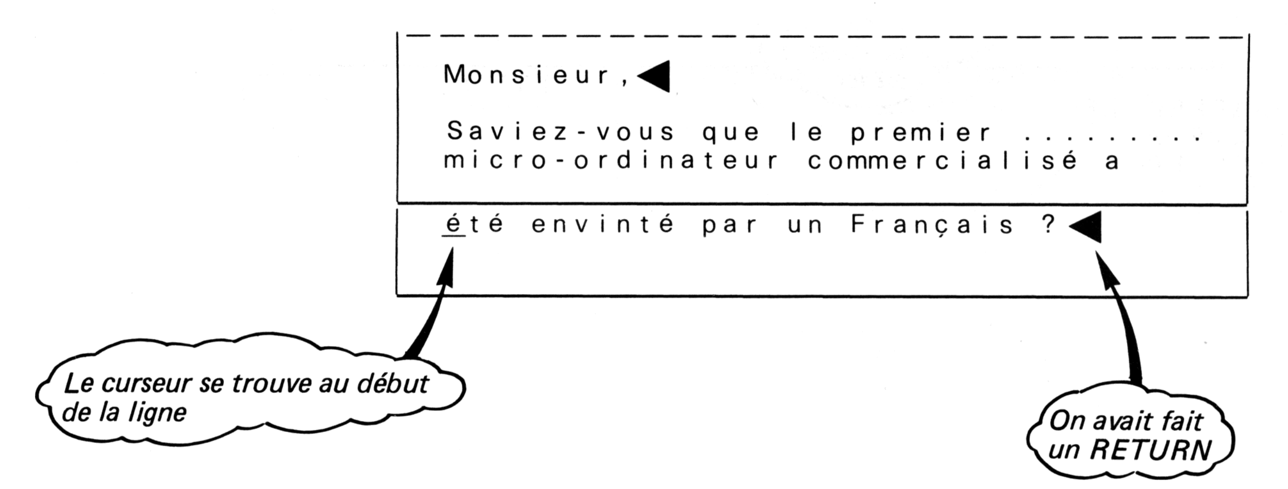


Remarquez qu'après 60 caractères, dans l'état actuel de nos margeurs, le texte frappé sur le « rouleau » passe d'office dans la zone texte. Il en va de même si vous frappez un retour chariot (RETURN) avant d'atteindre la limite fixée (60 caractères, ici).
Faites un RETURN. Les lignes sont désormais *au-dessus* du rouleau. Or, vous constatez que le mot *inventé* a été frappé par erreur est devenu *envinté :* il faut le corriger.

MODIFICATIONS OU CORRECTIONS

Les corrections et modifications ne peuvent porter que sur du texte présent dans le « rouleau noir ».

Pour ramener la ligne à corriger dans le rouleau, servez-vous des touches de droite organisées autour de la touche HOME et marquées ←, →, ↓ et ↑. Commencez par expérimenter les touches *verticales :* la touche ↑ fait *remonter* le rouleau dans le texte, alors que ↓ le fait redescendre ; tout se passe comme si elles servaient à promener le rouleau sur le texte. Vous obtenez :

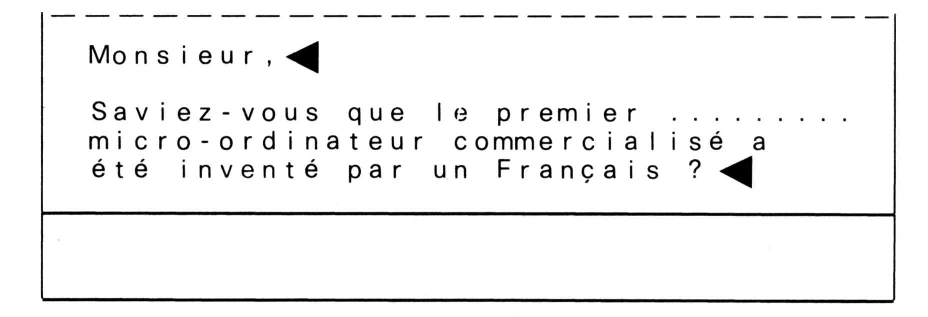


Cette fois, on va déplacer le curseur *horizontalement* grâce aux touches → et ← ; ne vous servez surtout pas de la barre d'espacement ou de la touche RETOUR/T/T qui, toutes deux, *effacent* les caractères.

GESTION DU CURSEUR

Les « touches de gestion du curseur », celles marquées de flèches (ainsi que celle marquée R, ou HOME) n'effacent pas les caractères qu'elles croisent.

Amenez donc le curseur sous le *e* qui commence *envinté* puis frappez directement un *i :* il prend sa place. Amenez ensuite le curseur sous le *i* et frappez un *e*, puis faites ↓ afin de constater que votre texte a regagné la zone texte exempt d'erreur.



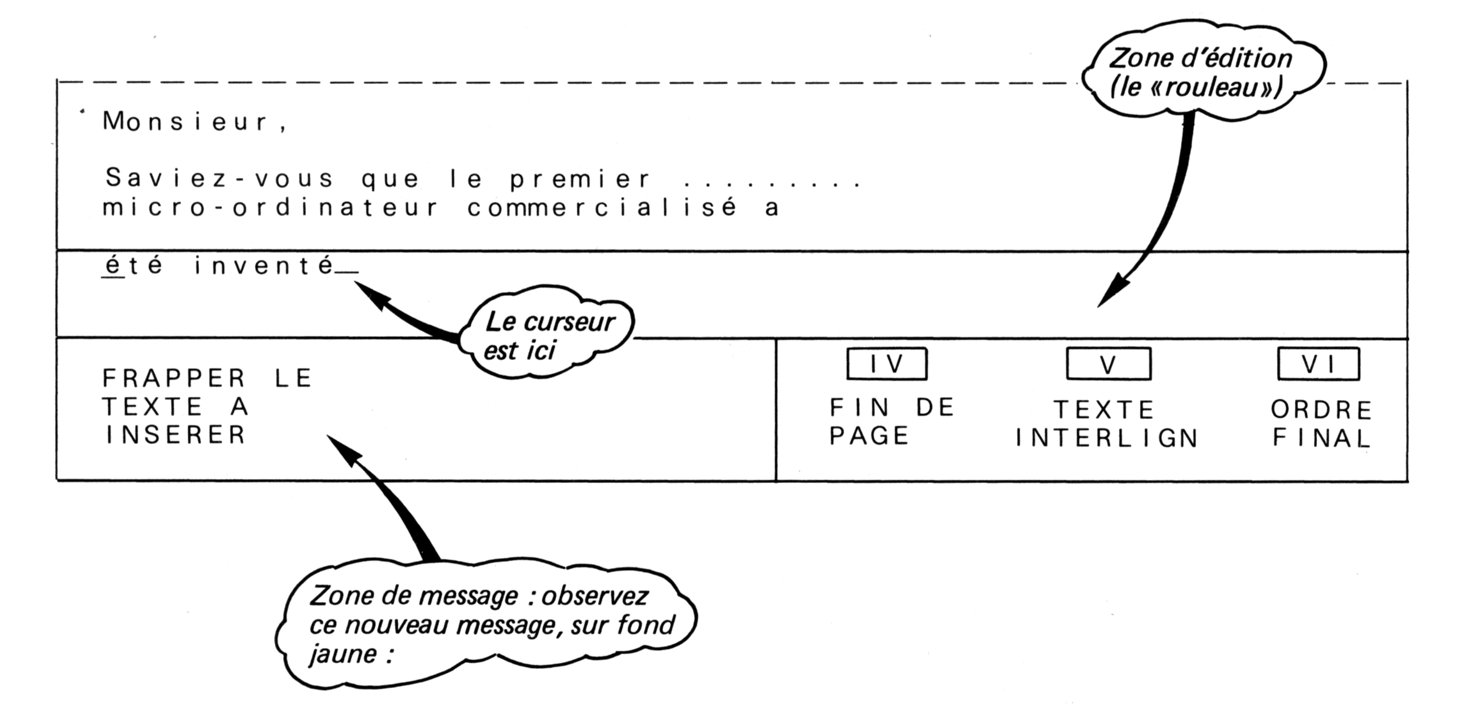
Ainsi avez-vous créé une entrée de lettre complète.

3. Insertion de texte

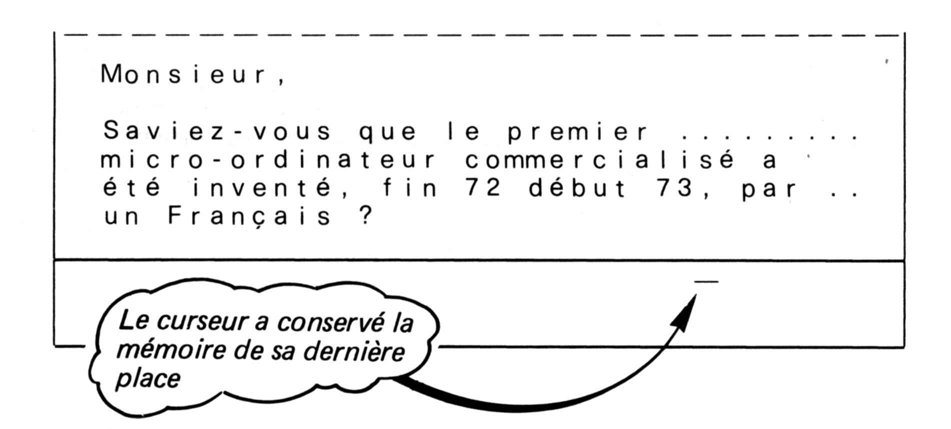
INSERTION

INSERER

Fort bien, mais on voudrait préciser la date et *insérer* cette précision dans la dernière ligne. Ramenez-la dans le « rouleau » qu'on appellera *zone d'édition* et amenez le curseur entre *inventé* et *par,* puis indiquez au traitement de texte (qu'on appellera aussi TT, en abrégé) que vous voulez réaliser une *insertion* à cet emplacement. Pour cela, frappez sur l'une des touches de fonction, celle marquée INSERER (dans le bloc supérieur de droite). Aussitôt, le texte qui suit le curseur disparaît (mais n'est pas perdu) et vous lisez :



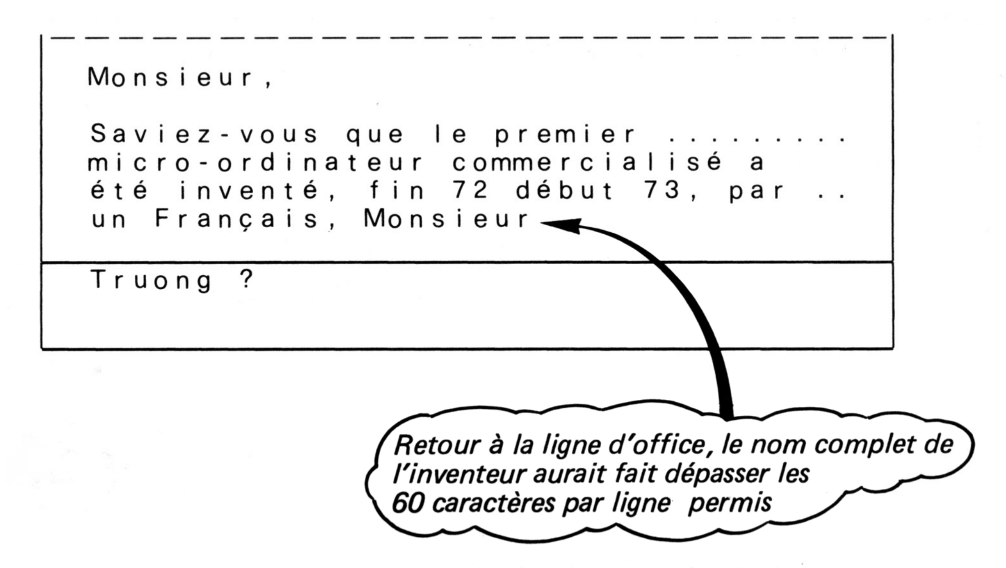
Puisqu'on vous invite à frapper le texte (message du bas), ajoutez « *, fin 1972 début 1973,* », puis, cela fait, signalez au TT que l'insertion est terminée en frappant sur la touche de dialogue numéro VI, laquelle se voit, en effet, attribuer cette fonction. Aussitôt, votre texte modifié est reprojeté dans la zone texte :



Ramenez la dernière ligne dans la plage d'édition (le « rouleau »), et vérifiez que :

- en déplaçant le curseur vers la droite avec →, vous entraînez le symbole de retour à la ligne vers la droite dès lors que le curseur est en dessous ;
- la commande ← ne ramène pas ce symbole ◀ vers la gauche ;
- pour le ramener à gauche, il faut effacer les espaces créés et, par conséquent : 1) placer le curseur sous ◀ et 2) frapper autant de fois que nécessaire sur RETURN/ESPACE.

Profitons-en pour terminer notre phrase. Ramenez la dernière ligne dans la zone d'édition, amenez le curseur tout simplement sous le symbole de retour à la ligne et supprimez le ? avec BACK-SPACE. Puis, frappez directement ce texte, qui décalera le curseur vers la droite au fur et à mesure de son entrée.

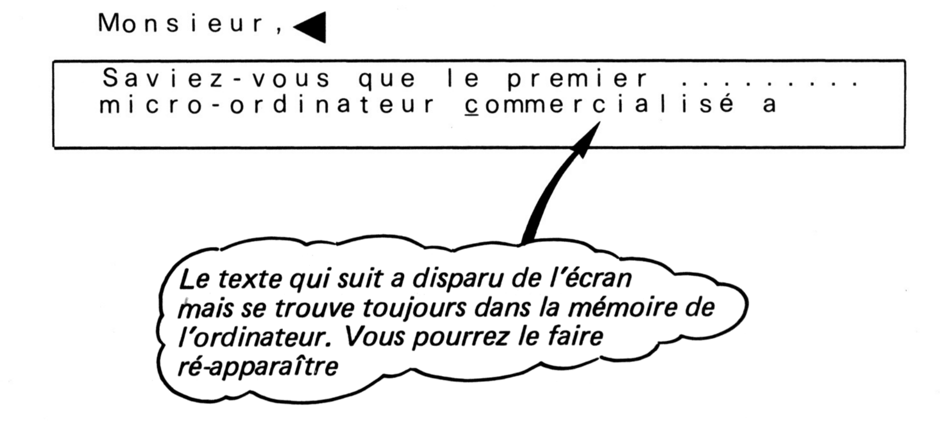


4. Suppression de texte

Puisque nous y sommes, examinons comment on supprime du texte ; par exemple, on va supprimer le mot *commercialisé.* Pour cela, il faut ramener sa ligne dans la zone d'édition, avec les flèches de gestion du curseur.

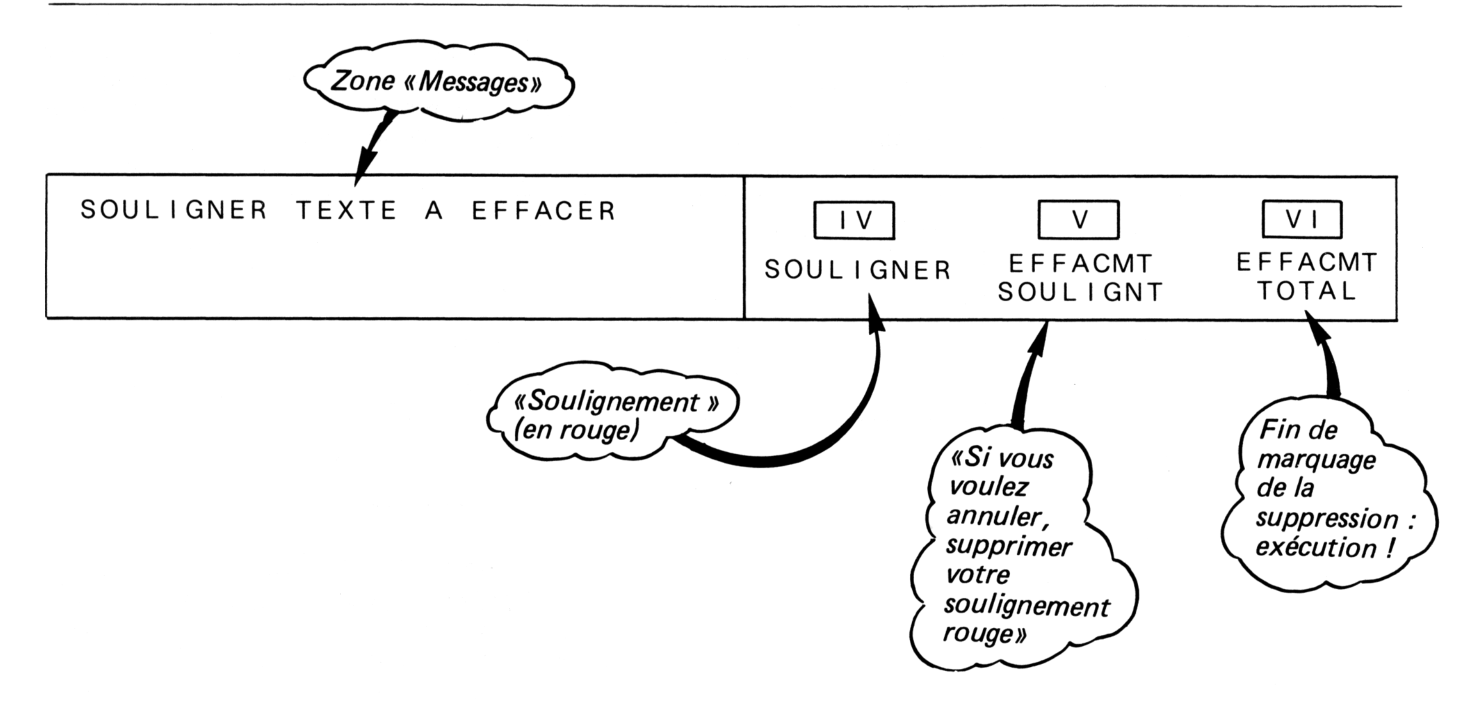
Remarquez qu'elles agissent non pas par « ligne affichée », ce qu'on appelle la « ligne physique », mais par « ligne logique », celle qui apparaîtra sur l'imprimante, donc de 60 caractères maximum.

Par conséquent, les déplacements se feront parfois deux lignes « physiques » à la fois. Vous devez obtenir, après avoir amené le curseur sous le *c* de *commercialisé :*



SUPPRESSION DE TEXTE
SUPPRESS

Maintenant, indiquez au TT (« traitement de texte ») que vous voulez *supprimer* quelque chose ; or, l'une des touches du bloc supérieur droit porte l'ordre SUPPRESS, pour *suppression.* Frappez-la et aussitôt, les messages du bas de l'écran se modifient ainsi :



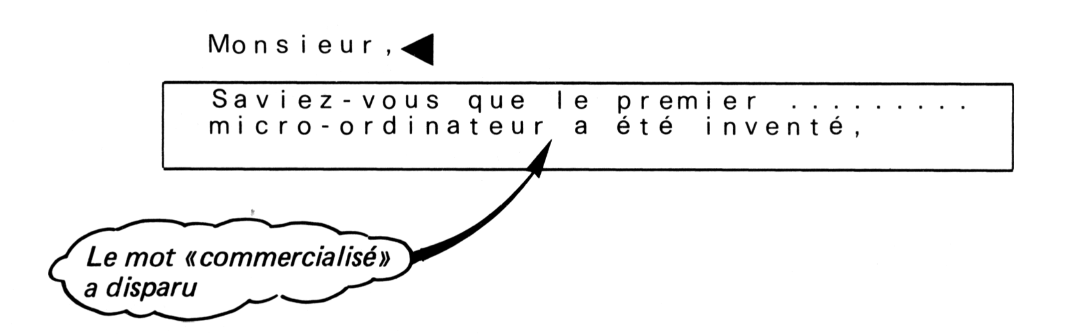
SOULIGNEMENT EN ROUGE
SOULIGNER

Vous êtes en mode suppression mais il vous faudra *souligner en rouge* le texte à supprimer. Pour démarrer ce soulignement rouge, frappez sur la touche de dialogue IV ; aussitôt, le curseur sous le *c* devient rouge et le message sous la touche IV (sur l'écran, toujours), indique ARRET SOULIGN., soit « Fin de soulignement en rouge » : pour retrouver le curseur blanc, il faudra refrapper la touche IV. Frappez sur la barre d'espacement qui, cette fois, n'efface rien mais souligne le mot *commercialisé* de rouge ; expérimentez aussi les touches V et IV à nouveau, par la même occasion :

- IV ramène le curseur blanc ;
- V vous permet d'agir comme avec une gomme avec la barre d'espacement pour effacer le souligné rouge ;
- notez aussi que RETOUR/ESPACE agit, ici, comme la barre d'espacement mais vers la gauche et peut vous servir à souligner un texte de rouge.

Rappelez-vous aussi que ces touches sont à répétition : maintenez-les enfoncées pour répéter automatiquement et vite l'opération.

Le mot commercialisé, plus l'espace suivant étant soulignés de rouge, frappez sur VI pour commander l'exécution. Aussitôt, vous sortez du mode « suppression » et vous obtenez :



Vous pouvez, dès lors, faire « remonter » votre texte dans la zone de texte :

```
Monsieur, ◀

Saviez-vous que le premier .........
micro-ordinateur commercialisé a
fin 72 début 73, par un Français, ..
Monsieur Truong ?◀
```

La suppression a ainsi été exécutée et le texte « refermé ».

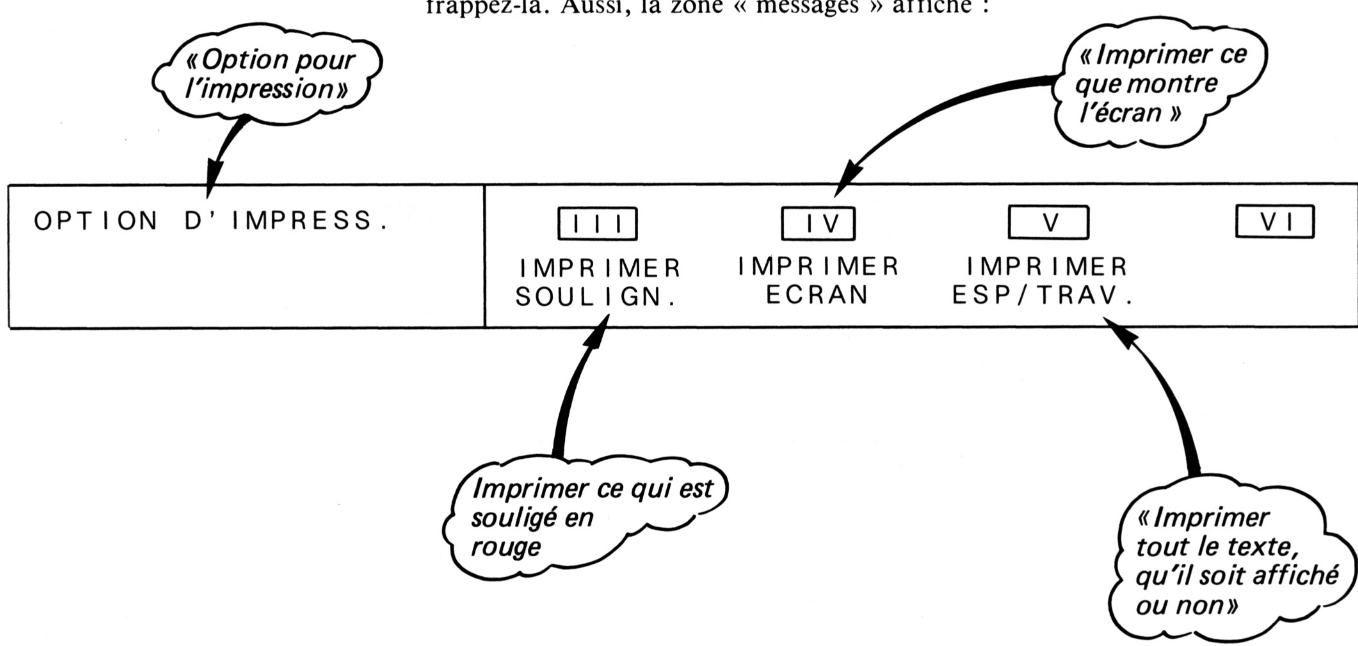
Si, en cours de route, vous commettez une erreur de commande, annulez ce que vous avez commandé en frappant sur ESCAPE/WP. La zone de messages du bas de l'écran vous ramènera les messages de départ.

RETOUR/T/T

Cette touche RETOUR/T/T nous avait servi, en début de session, à entrer en mode « traitement de texte ». Parce qu'on s'y trouve, elle constitue maintenant une touche d'échappement de « retour ».

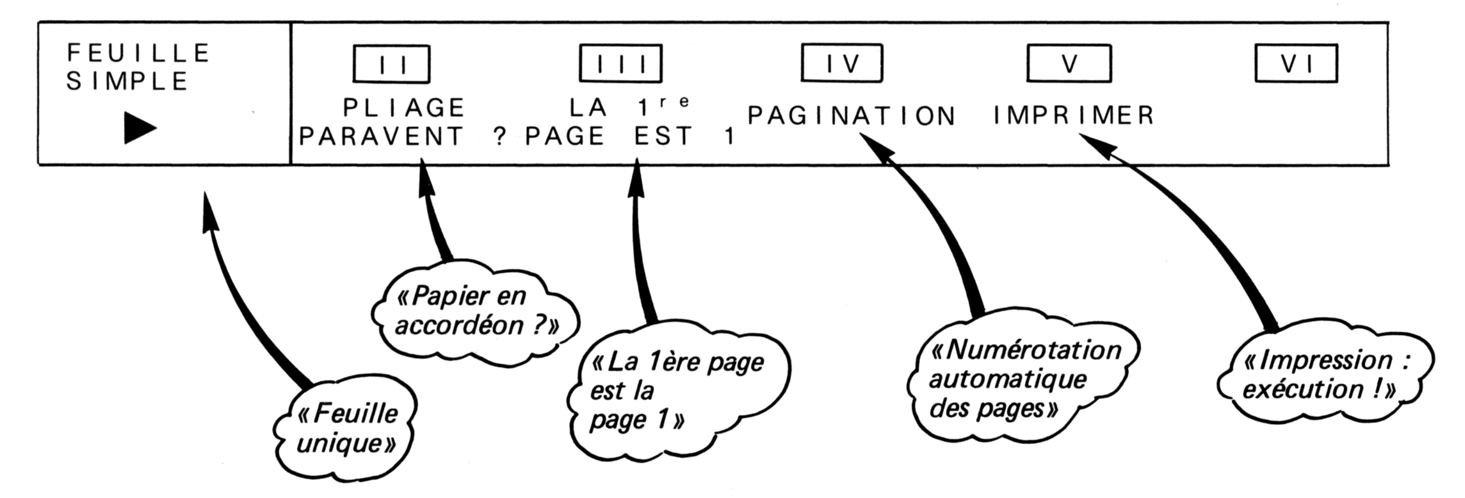
5. Sortie sur imprimante

Supposons qu'on veuille, maintenant, « sortir » ce début de lettre sur papier. Introduisez une feuille de papier dans votre imprimante, puis repérez une autre des touches de fonctions du bloc supérieur de droite, celle marquée IMPRESSION ; frappez-la. Aussi, la zone « messages » affiche :



IMPRIMER

Puisqu'on veut imprimer ce qui est affiché par l'écran, frappons sur la touche IV. A nouveau, les messages se modifient et se font plus précis :

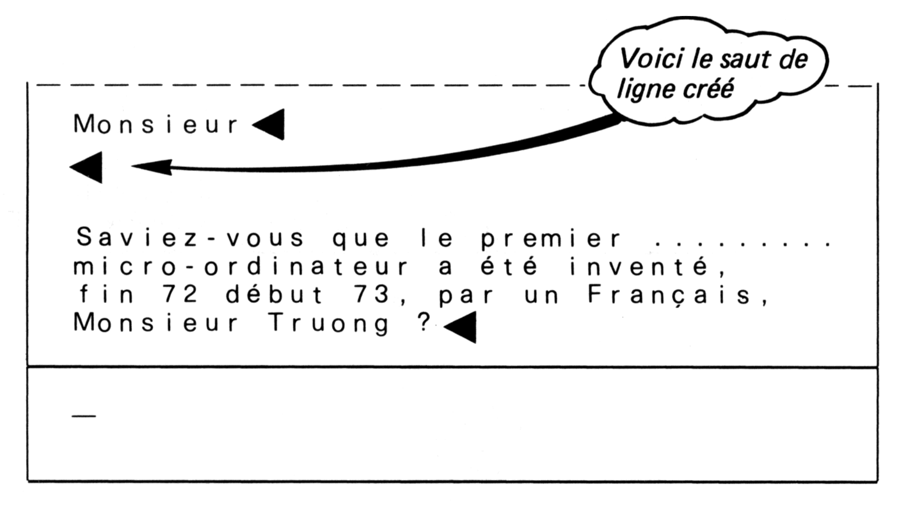


Frappez donc sur V pour déclencher l'impression, qui démarre aussitôt. Notez que le marquage de cette même touche devient « ARRET IMPRESS. » : si vous la refrappez, l'impression stoppe. L'auteur a obtenu :

```
Monsieur,
Saviez-vous que le premier micro-ordinateur a été inventé,
fin 72 début 73, par un Français, Monsieur Truong ?
```

ORDRE FINAL

Tout le texte apparaît bien sur des lignes de 60 caractères mais malheureusement, il n'y a pas eu de saut de ligne entre *Monsieur* et la suite ; en effet, la ligne *vide* qui apparaît à l'écran termine une *unique* ligne de 60 caractères. Créons un saut de ligne et pour cela, rappelons les deux lignes de texte dans la zone d'édition ; placez le curseur sous le S de « Saviez » et commandez une *insertion* en frappant la touche INSERER. Lorsque la zone de messages vous commande « FRAPPEZ LE TEXTE A INSERER » , frappez uniquement un RETURN (↵). Frappez ensuite sur VI (fonction « ORDRE FINAL » : *c'est fait !*), remontez le texte sur l'écran ; vous obtenez :

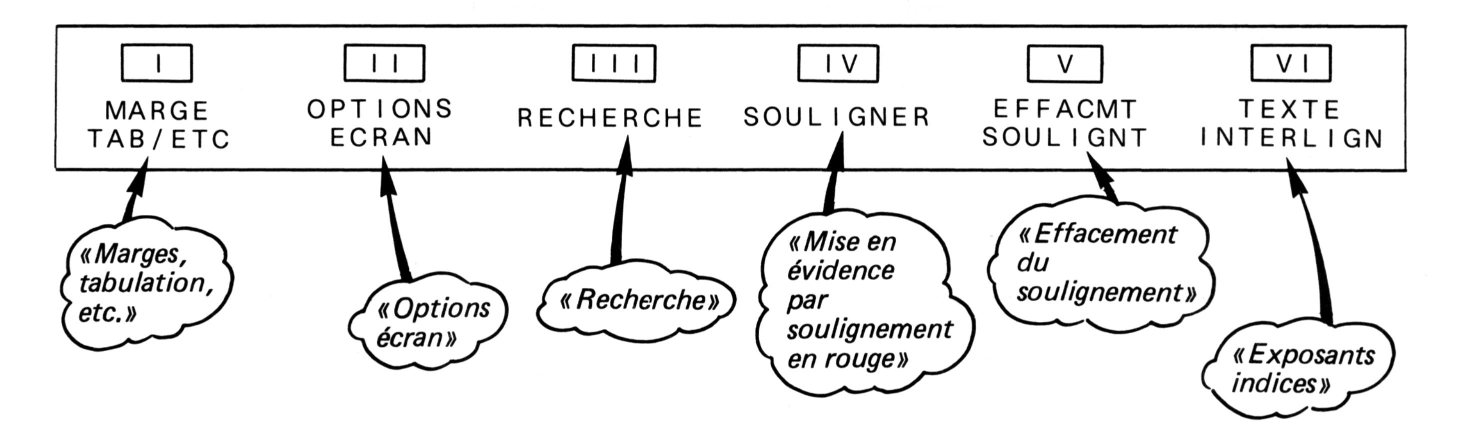


Relancez l'impression avec IMPRESSION et vous obtiendrez, cette fois :

```
Monsieur,

Saviez-vous que le premier micro-ordinateur a eté inventé,
fin 72 début 73, par un Français, Monsieur Truong ?
```

Si vous vouliez n'imprimer que les deux lignes du texte lui-même, soulignez-le en rouge et utilisez l'option correspondante. Le processus est le suivant ; en fonctionnement normal (frappe), les messages montrent :



Frappez donc sur la touche de dialogue IV, afin de mettre en service le soulignement.

Cette fois, exécutez le soulignement uniquement avec les touches de gestion du curseur → et ← car la barre d'espacement ou RETOUR/ESPACE effaceraient les caractères.

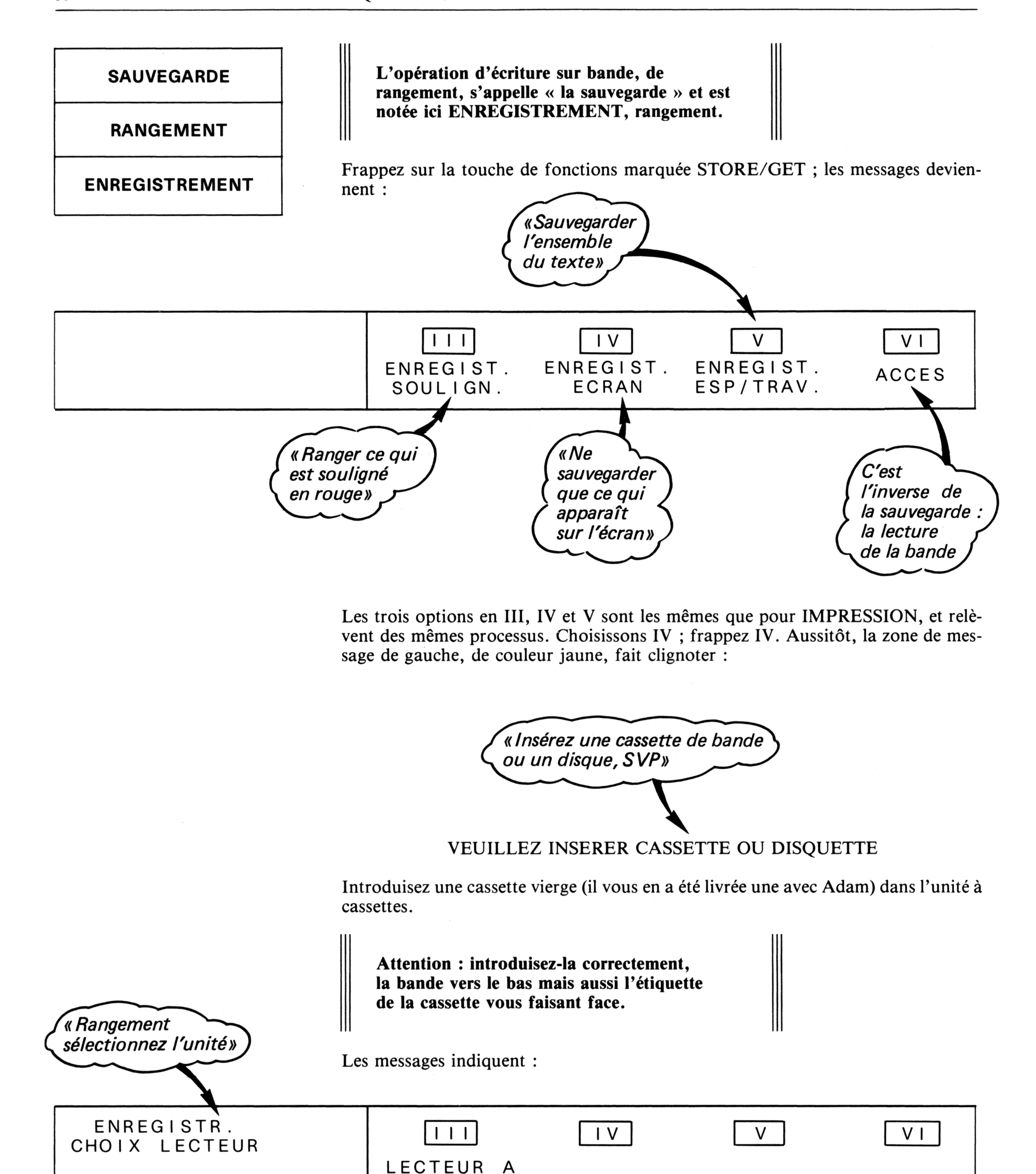
Amenez les lignes successives à souligner dans la zone d'édition, soulignez-les en rouge, puis refrappez sur la touche IV qui indiquait, dès sa première frappe, ARRET SOULIGN.
Dès lors, frappez sur la touche IMPRESSION, puis lorsque le message OPTIONS D'IMPRESS. apparaît (comme précédemment), frappez sur III car c'est la touche qui commande l'impression du texte souligné en rouge uniquement (« IMPRIMER SOULIGN. »). Aux messages suivants, répondez comme vous l'avez déjà fait en frappant sur V (« IMPRIMER »). L'impression se fait aussitôt mais cette fois, seul le texte souligné est reproduit :

```
Saviez-vous que le premier micro-ordinateur a été inventé,
fin 72 début 73, par un Français, Monsieur Truong ?
```

Pour effacer le soulignement sur l'écran, comme avec une gomme, frappez d'abord sur la touche de dialogue V (« EFFACMT SOULIGNT ») et effacez en vous servant des touches de gestion du curseur, puis annulez cette commande en refrappant V devenue, entre temps, « ARRET EFFACMT » soit « fin de l'effacement ».
La troisième option d'OPTIONS D'IMPRESS. est notée IMPRIMER ESP/TRAV. (touche V). Cette option vous permet d'imprimer tout le texte en mémoire et non plus seulement ce qui apparaît sur l'écran.

6. Sauvegarde sur cassette et chargement

Peut-être voulez-vous ranger votre texte sur bande magnétique avant de mettre Adam hors tension ? En effet, rappelez-vous que les *mémoires internes vives* sont volatiles : lorsque vous mettrez la machine hors tension, elles vont perdre ce texte. Par contre, la bande magnétique le conservera si vous l'y avez enregistré, et vous le restituera à la demande.



L'opération d'écriture sur bande, de rangement, s'appelle « la sauvegarde » et est notée ici ENREGISTREMENT, rangement.

Frappez sur la touche de fonctions marquée STORE/GET ; les messages deviennent :

Les trois options en III, IV et V sont les mêmes que pour IMPRESSION, et relèvent des mêmes processus. Choisissons IV ; frappez IV. Aussitôt, la zone de message de gauche, de couleur jaune, fait clignoter :

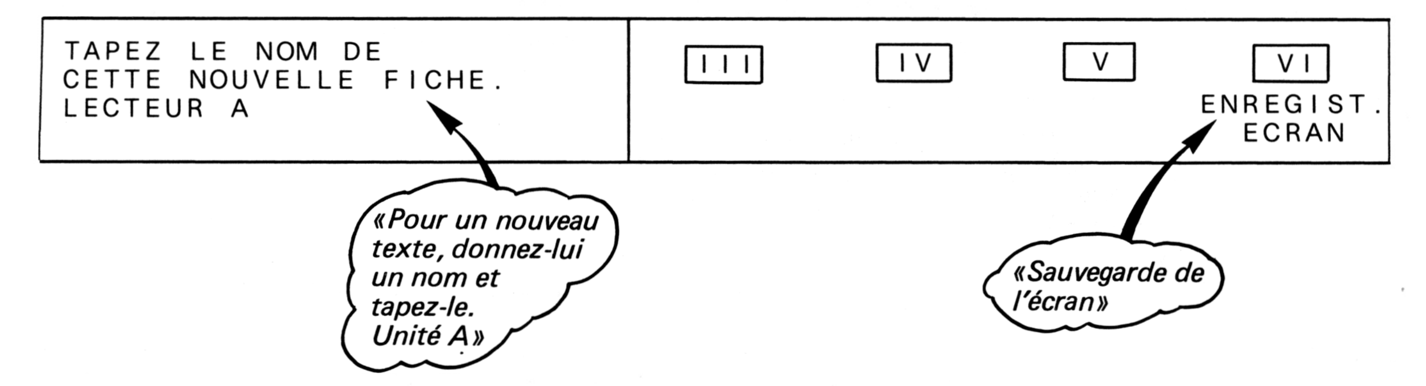
VEUILLEZ INSERER CASSETTE OU DISQUETTE

Introduisez une cassette vierge (il vous en a été livrée une avec Adam) dans l'unité à cassettes.

Attention : introduisez-la correctement, la bande vers le bas mais aussi l'étiquette de la cassette vous faisant face.

Les messages indiquent :

L'unité A est le nom de code de votre unité à cassettes. Frappez alors sur la touche de dialogue III ; les messages indiquent :



FICHIER

La règle veut, en effet, qu'on décerne un nom à tout ce qui est créé. Un texte créé, quelle que soit sa longueur, s'appelle *un fichier* dans la langue des informaticiens Appelons notre « fichier » : *Invention.* Frappez *Invention* qui s'inscrit dans la zone d'édition, puis frappez sur VI. Aussitôt, l'unité à cassettes s'anime et la sauvegarde se fait, ce qui va prendre quelques secondes. Pendant ce temps, la zone messages affiche sur fond jaune :



UN INSTANT : FICHE EN COURS D'ENREGISTREMENT

A l'issue de l'opération, l'écran restitue les messages d'origine.
Dès lors, faites un RESET COMPUTER en actionnant le poussoir situé sur la face supérieure de l'unité centrale. Ce faisant, vous effacez la mémoire de l'ordinateur et l'écran ; votre texte est perdu ; le résultat est semblable si vous débranchez l'ordinateur puis le remettez en service. On va récupérer ce « fichier ».

RESTITUTION

LECTURE

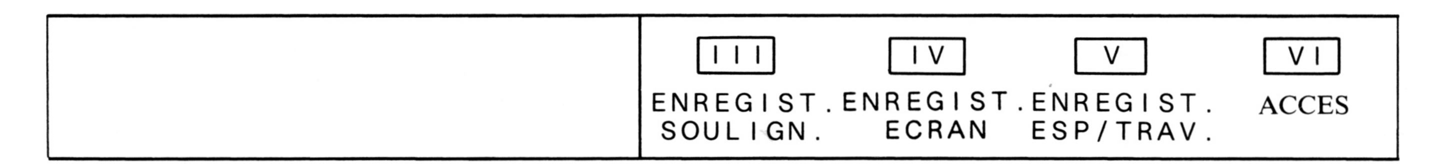
CHARGEMENT

ACCES

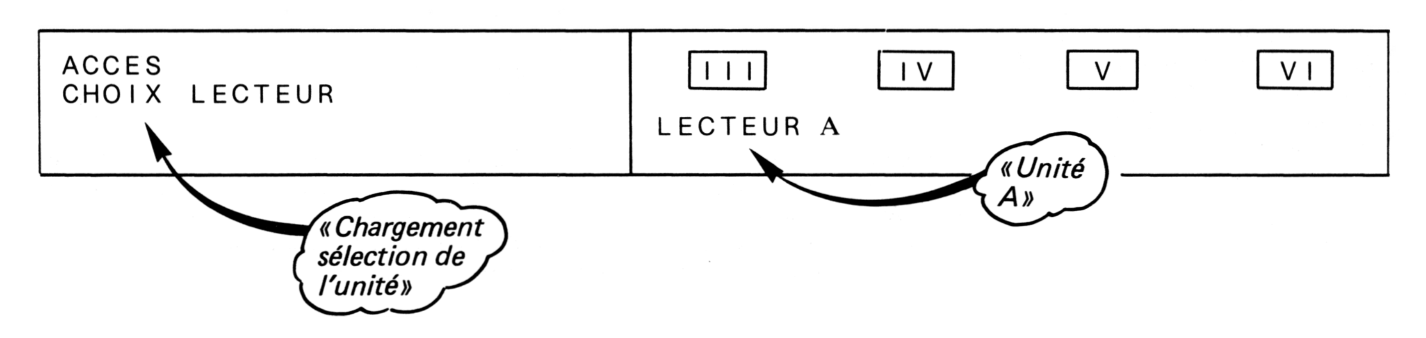
Repartez en traitement de texte en frappant RETOUR/T/T, puis frappez sur la touche ENREGIS./ACCES.

La restitution d'un « fichier », son appel, se nomme ACCES. En langage d'informaticien, on dira qu'on « charge » les mémoires centrales et cette opération s'appellera « chargement ».

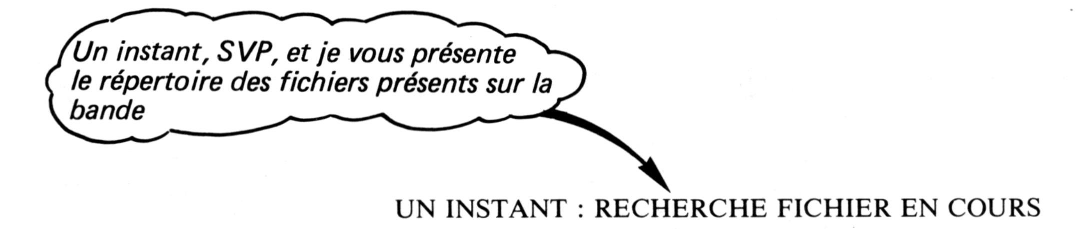
Vous retrouvez les mêmes messages sur les lignes du bas :



Cette fois, frappez sur VI, pour ACCES. Les messages deviennent :



En effet, l'ordinateur peut toujours supposer que vous installerez d'autres unités à cassettes (ou à disquettes). N'en ayant découvert qu'une, il vous la propose. Frappez sur III ; le message sur fond jaune indique un instant :



UN INSTANT : RECHERCHE FICHIER EN COURS

FICHIER

En effet, le traitement de texte commence par lire la « table des matières », appelée encore ici FICHIER, et qui regroupe tous les noms de tous les fichiers que vous avez sauvegardés antérieurement. Il affiche ensuite ce « fichier central » ; en vous servant des touches de gestion du curseur, vous amènerez un pointeur en face du nom du fichier sélectionné, ce qui donnera :

«Répertoire des fichiers»

C'est le nôtre : il est d'ailleurs le seul.

FICHIER

Invention

en rose bonbon

Pointeur

Les rectangles réservés aux noms des fichiers sont en bleu

SELECTION D'UN FICHIER A PARTIR
DU LECTEUR A
UTILISEZ LES TOUCHES A FLECHES

IV CHOIX LECTEUR

V SECOND FICHIER

VI ACCES FICHIER

«Sélection d'unité»

C'est une liste de noms en secours

«Appel du fichier» (pour chargement)

«Sélectionnez un fichier de l'unité A. Servez-vous des touches de gestion du curseur»

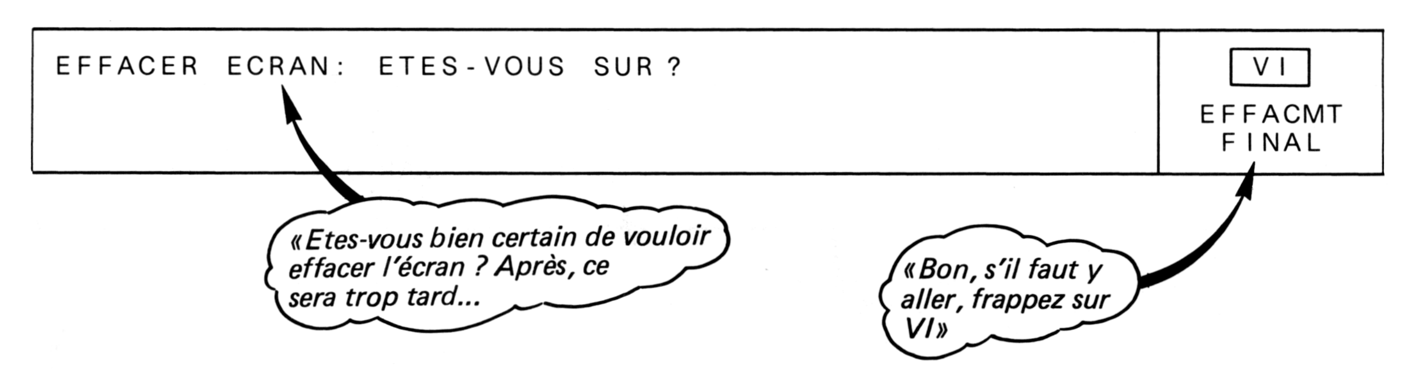
Le pointeur se trouvant en face de *Invention,* frappez sur VI. L'unité à cassettes s'anime un instant tandis qu'un message vous invitant à prendre patience s'affiche, puis l'écran revient au traitement de texte et la première ligne de votre fichier est affichée ; rassurez-vous, il se trouve totalement en mémoire : vous pouvez le faire défiler sur l'écran avec les touches de gestion du curseur.

Ré-affichez-le donc en totalité : nous allons vous montrer comment effacer l'écran. Pour cela, frappez sur la touche de fonction marquée EFFACER. Les messages indiquent :



EFFACER

Frappez sur V, dans notre cas ; il vient (admirez la prudence du traitement de texte !) :



Si vous frappez sur VI, le texte affiché s'efface aussitôt et vous retrouvez un bel écran neuf. Si vous changiez d'idée, il vous faudrait frapper sur RETOUR/T/T. Si vous aviez demandé l'effacement complet de « l'espace de travail », c'est-à-dire de tout ce qui est en mémoire, tout votre texte aurait été perdu, en mémoire centrale, bien entendu. Car celui qui se trouve en cassette n'est pas affecté.
Sauf si... *après* l'effacement et saisi de remords, vous voulez donner un contre-ordre. C'est encore possible (et c'est même assez exceptionnel) : frappez sur la touche de fonction ERREUR.

CONTRE-ORDRE

ERREUR

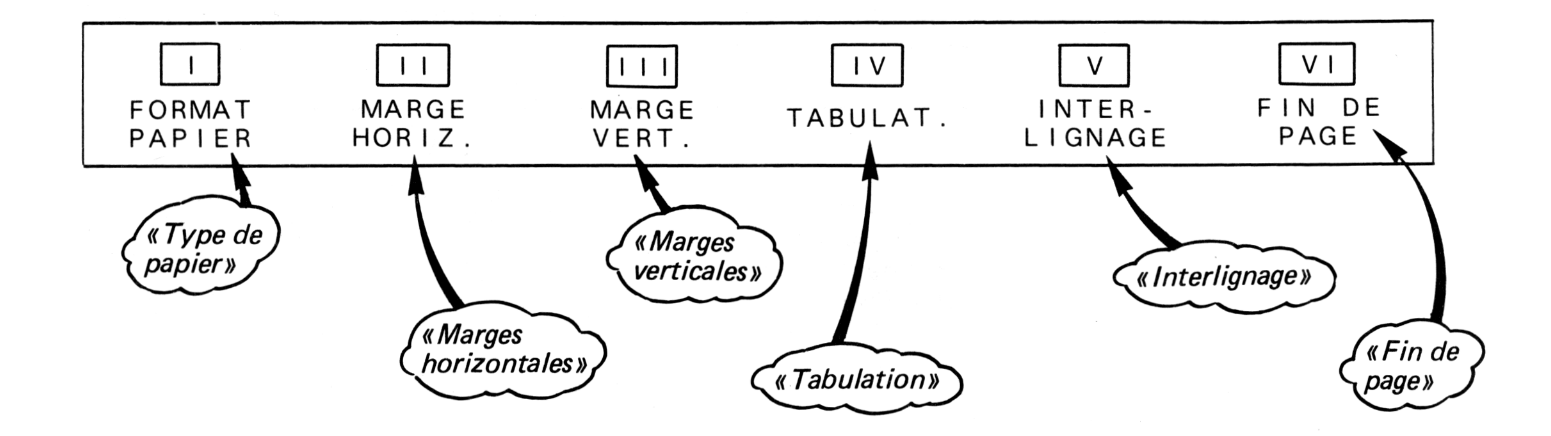
ERREUR (« contre-ordre ») annule l'effet destructif d'un ordre précédent : EFFACER, ou encore SUPPRESS. ou RETOUR/ESPACE.

Vous éprouverez la joie de voir ré-apparaître votre texte dans son intégralité.

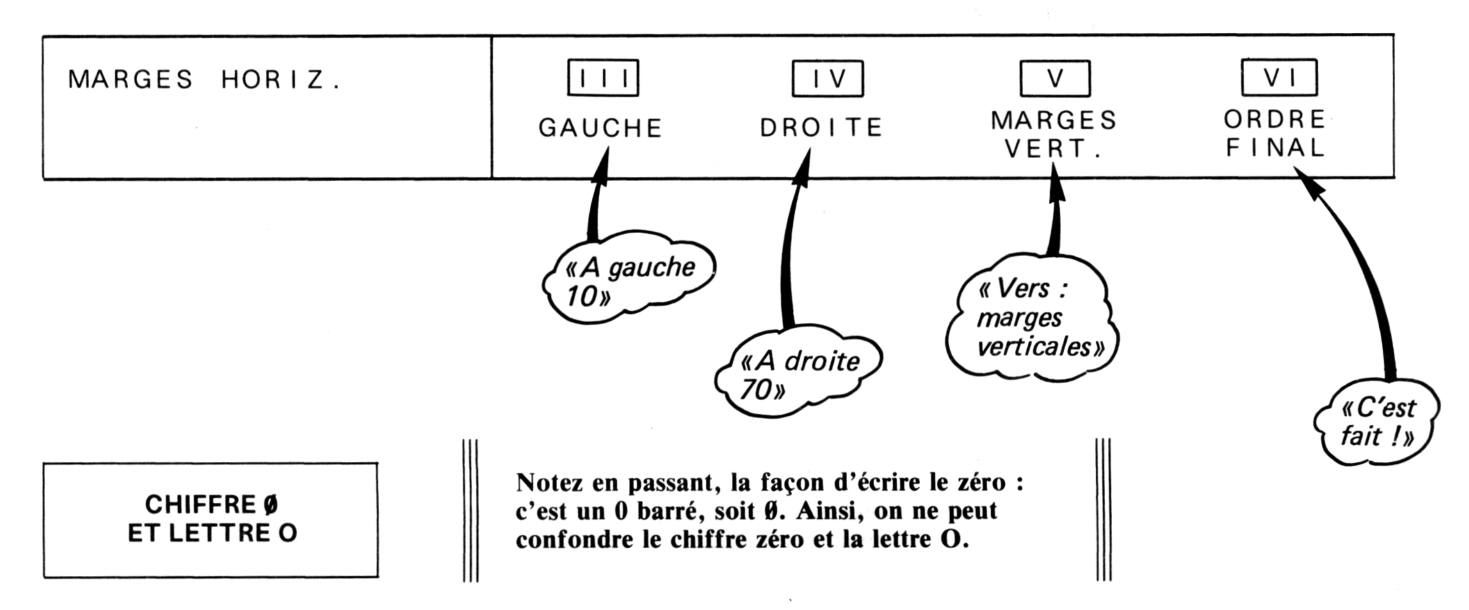
7. Marges

MARGE

Vous devez, maintenant, mieux comprendre l'action du traitement de texte, aussi va-t-on accélérer le mouvement afin d'en examiner diverses autres possibilités (rappelez-vous que le manuel livré avec Adam ne vous fait grâce d'aucune d'elles). Commençons par changer les marges. Dès les messages d'origine, frappez sur la touche de dialogue I marquée MARGE/TAB/ETC. Les messages deviennent :



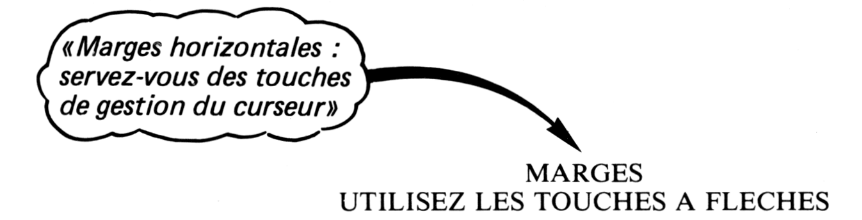
Démarrons par les marges horizontales (nous n'avons pas créé assez de texte pour les *verticales,* haut et bas de page, mais le processus resterait le même). Frappez sur II ; les messages indiquent :



CHIFFRE Ø ET LETTRE O

Notez en passant, la façon d'écrire le zéro : c'est un 0 barré, soit Ø. Ainsi, on ne peut confondre le chiffre zéro et la lettre O.

Cette façon de procéder est très générale en informatique. Mais ajustons d'abord la marge gauche, fixée au départ à 10 caractères par le traitement de texte ; on va la porter à 20. Frappez sur III ; seul le message sur fond jaune change ; il devient :



Grâce aux touches fléchées → et ←, vous allez *incrémenter* ou *décrémenter* le nombre 10. Portez-le à 20 avec → et observez le taquet de tabulation horizontal, sur le trait noir supérieur du haut de l'écran : ce taquet *rouge* se déplace vers la droite.

Puis frappez sur IV pour *marge droite ;* on va la porter à 65 au lieu de 70, ce qui laissera une marge de 15 caractères. Cela fait, frappez ORDRE FINAL (« c'est fini ! ») puis faites un essai d'impression du texte qui a été créé (rechargez-le en mémoire pour la circonstance, à partir de la cassette et comme vous devez savoir le faire). Voici le résultat :

```
Monsieur,

Saviez-vous que le premier micro-ordinateur a
été inventé, fin 72 debut 73, par un Français,
Monsieur Truong ?
```

Cette fois, les lignes sont de 45 caractères. Si vous faites réapparaître le texte sur l'écran, vous constaterez qu'il a, lui aussi, été réorganisé. Essayez encore avec des marges de 30 et 60 ; vous obtiendrez :

```
Monsieur,

Saviez-vous que le premier
micro-ordinateur a été inventé,
fin 72 debut 73, par un
Français, Monsieur Truong ?
```

TABULATIONS

Vous pourriez également poser des taquets de tabulation avec TAB, toujours en suivant les indications du traitement de texte. Notez, à ce propos, que la touche de tabulation, TAB, est située à gauche du clavier ; le TT (traitement de texte) fixe les tabulations d'origine de 5 en 5 caractères : si vous pressez TAB, le curseur se déplace sur la prochaine butée de tabulation, comme avec une machine à écrire.

HOME OU R

Supposons, maintenant, que le texte ci-dessus se trouve dans la zone texte : vous voulez le reprendre à partir du début. Au lieu de vous servir des touches fléchées, frappez uniquement sur la touche qui se trouve à leur centre et est marquée HOME mais aussi R. Aussitôt, la première ligne « Monsieur, » apparaît dans la zone d'édition, le curseur se positionnant sous le *M.* N'est-ce pas plus rapide ainsi ?

8. Recherches et remplacements

Selon les mêmes principes, vous disposez de multiples autres fonctions. Dans tous les cas, utilisez les touches nommément marquées du clavier (en gris moyen) et laissez-vous guider par les « menus » qui apparaissent sur l'écran et qui affectent un rôle précis aux touches de dialogue.

RECHERCHE

Ainsi, vous pouvez rechercher automatiquement un mot ou une expression (32 caractères au maximum) avec la commande RECHERCHE *(« recherche »)* disponible sur la ligne de commande du bas de l'écran dès la mise en service du traitement de texte. Si vous la frappez (touche III), la ligne de commande vous demande ce que vous recherchez (RECHERCHE DE) ; frappez alors le mot ou l'expression recherchée, puis DEMARRER RECHERCHE ; le traitement de texte placera le curseur sur le début du texte recherché. Vous pourriez poursuivre plus avant dans le texte en frappant ensuite RECHERCHE SUIVANTE *(« recherche de l'occurence suivante »)* et répéter cette opération jusqu'au message TEXTE INTROUV. *(« Je n'ai pas trouvé ce texte ! »).*

Avant toute recherche, placez le curseur *au début* du texte car c'est à partir de sa position que la recherche débutera.

Eventuellement, frappez sur la touche R afin de le ramener tout à son début.

Le texte trouvé, vous pouvez procéder à sa modification, ou même à son remplacement avec l'ordre CHANGER affecté alors à la touche V ; l'écran vous demandera

CHANGER

par quoi remplacer le mot ou l'expression (par exemple, le mot *francs* en mot *dollars),* puis s'il faut remplacer *d'office* toutes les apparitions du mot *francs* par *dollars*.
Vous pourrez aussi :

• Modifier la couleur du fond de l'écran et au lieu du bleu, placer du blanc, du vert, du noir, du gris ou... revenir au bleu.

• Réduire les sons aux seuls messages destinés à attirer votre attention, ou les supprimer totalement.

• Marquer la fin de chaque « page » de texte avec un marqueur spécial (frappez d'abord sur MARGE/TAB/ETC puis sur FIN DE PAGE). L'impression s'arrêtera page par page ou, si vous employez du papier en accordéon, « tournera » la page pour se poursuivre.

• Numéroter les pages automatiquement.

• Déplacer un mot ou un paragraphe complet sans avoir à le refrapper (avec la touche DEPLACT/COPIER d'abord).

• Dupliquer des passages.

• Modifier les tabulations, fixées d'office de 5 en 5 caractères. Ce sera bien utile pour écrire des tableaux ou des colonnes.

• Dans ce cas, vous pourrez même supprimer le « rouleau d'entraînement » sur l'écran en frappant sur OPTIONS ECRAN, puis sur ECRAN MOBILE. Dans ce cas, le texte frappé se présentera tel qu'il sera sur votre feuille de papier et vous « promènerez » l'écran, comme une « fenêtre » de 36 caractères de large, sur des pages pouvant atteindre 80 caractères.

Bref, vous disposez de dizaines de possibilités qui font à la fois de ce traitement de texte un outil très utile et très efficace si vous avez des textes à rédiger, mais aussi, peut-être l'un des plus simples à apprendre qu'il ait été donné à l'auteur de rencontrer.

Questions sur le chapitre II

Vous connaissez maintenant notre jeu des questions. en voici quelques-unes, importantes, sur le traitement de texte.

1. Pour entrer en mode traitement de texte, il faut :
   - ☐ Utiliser une cassette-programme spéciale comportant le programme de traitement de texte
   - ☐ Frapper sur RETOUR/T/T après la mise en service de l'ordinateur
2. Pour effacer un caractère précédent, il faut frapper sur :
   - ☐ RETOUR/ESPACE
   - ☐ SUPPRESS.
3. Pour annuler une commande non encore exécutée, il faut frapper sur :
   - ☐ RETOUR/T/T
   - ☐ ERREUR
4. L'opération d'impression se dit :
   - ☐ PAPIER
   - ☐ IMPRESSION
5. L'opération d'enregistrement d'un texte sur cassette se dit :
   - ☐ ENREGIST.
   - ☐ SAUVEGARD.
6. La lecture d'une cassette pour charger un texte dans l'ordinateur se dit :
   - ☐ CHARGMT
   - ☐ ACCES

Solutions page 38

Réponses aux questions sur le chapitre I

1. L'unité centrale, c'est le bloc comprenant l'ordinateur proprement dit.

2. Le clavier AZERTY est le clavier aux normes françaises.

3. Pour passer en majuscules, il faut actionner SHIFT, la laisser enfoncée, puis frapper sur le caractère qui doit apparaître en majuscule. La touche RETURN (ou ↵) provoque un retour à la ligne.

4. RESET (COMPUTER) provoque une remise à l'état initial du système.

5. Un menu, c'est un choix entre diverses options.

6. Il faut toujours mettre l'ordinateur hors tension pour insérer ou extraire une cartouche.

7. Il ne doit jamais se trouver une cassette en place dans son unité à cassettes lorsqu'on met l'ordinateur en service ou hors service.

Allons, combien de réponses justes ? Sept sur sept ? Tous nos compliments. De 5 à 7 ? Mais c'est très bien encore. De 3 à 5 ? N'hésitez pas à revoir les points traités dans ce chapitre. Moins de 3 ? Mais au fait, à quoi pensiez-vous en lisant ce chapitre ?

Réponses des exercices sur le chapitre II

1. Adam fait exception : il est l'un des très rares ordinateurs à disposer d'un traitement de texte *incorporé*. Il vous suffit de frapper sur la touche RETOUR/T/T après sa mise en service. Il n'y a d'ailleurs pas de cassette de traitement de texte !

2. RETOUR/ESPACE, parce que c'est plus simple. (SUPPRESS. vous servira à supprimer ensuite des mots ou des passages.)

3. Si elle est en cours, annulez une commande avec RETOUR/T/T. (L'ordre ERREUR annule un effacement malencontreux qui vient tout juste d'être exécuté.)

4. L'impression se dit IMPRIMER.

5. L'enregistrement sur cassette se dit ENREGIST. avec ce traitement de texte.

6. La lecture d'une cassette se dit ACCES avec ce traitement de texte.

Voyons, faites le compte... Zéro faute, nous l'espérons ? N'hésitez pas à pratiquer intensivement le traitement de texte car c'est l'unique méthode pour bien l'assimiler. A propos, notre histoire d'invention du micro-ordinateur, elle est parfaitement authentique : nous avons battu les Américains, ce qui est prestigieux.

CHAPITRE III

LE DIALOGUE AVEC L'ORDINATEUR EN MODE DIRECT

Vous savez désormais exploiter des cartouches ou des cassettes pré-enregistrées. Vous êtes en passe de devenir un expert en traitement de texte. Que pouvez-vous apprendre de plus ? Tout simplement, à « programmer » vous-même votre ordinateur, à lui fournir des séquences d'ordres qui lui feront exécuter des programmes nés de votre intelligence et de votre imagination. Pour cela, nous allons découvrir un « langage » de programmation, le Basic, et l'employer tout d'abord en « mode direct » : vous donnez un ordre à l'ordinateur et il l'exécute aussitôt.

Principaux thèmes étudiés

Instructions	PR#1	Division (/)
Langage	PR#0	Point décimal
Basic	Suppression d'un caractère	Virgule décimale
Applesoft	RETOUR/ESPACE et ←	Puissances
SmartBasic	Recopie de caractères	Racine carrée
Messages d'erreur	Effacement d'un caractère sur l'écran	SQR
Format de l'écran	CONTROLE/O	Ensemble numérique
CONTROLE	Annulation d'une ligne	Chaîne de caractères
Effacement de l'écran	CONTROLE/X	Symbole « dollars »
CONTROLE/L	? pour PRINT	Variable
PRINT	Opérations sur les chiffres	Constante
RETURN ou ↵	Point-virgule dans un PRINT	Symbole d'affectation =
Guillemets	Multiplication (*)	Noms des variables
Impression de l'écran		Mots réservés
CONTROLE/P		Concaténation de chaînes

1. Qu'est-ce que le Basic ?

Un ordinateur se distingue des autres types de machines par le fait qu'on peut lui déléguer des « fonctions intellectuelles ». Il sait exécuter énormément de choses si on les lui « explique » correctement : poser une fusée sur la lune, faire du traitement de texte, établir une facture, jouer aux échecs, vous enseigner la géométrie et le calendrier... Pour cela, il faut lui fournir des *instructions* précises développant pas à pas ce qu'il doit faire *dans un langage* qu'il soit capable de comprendre ! Là réside l'une des difficultés majeures.

INSTRUCTIONS

LANGAGE

En effet, ce langage doit pouvoir, *aussi,* être compris par l'homme. Le français courant ou l'anglais courant sont, eux, trop compliqués encore pour la machine, aussi a-t-on recherché des compromis : un anglais simplifié à l'extrême de quelques dizaines de mots a servi de base au langage de programmation le plus utilisé actuellement, le « Basic ». Rassurez-vous : même si vous êtes rebelle aux langues étrangères, vous n'éprouverez aucune difficulté à assimiler les quelques mots anglais nécessaires qui vous apparaîtront d'ailleurs comme des codes.

Développé aux Etats-Unis au cours des années 60, le Basic doit son nom aux premières lettres de l'expression *« Beginners All purpose Symbolic Instruction Code »,* soit à peu près *code d'instructions symboliques tous usages pour débutants.*

BASIC

Depuis son invention, le Basic a fait souche : il a crû et s'est multiplié. Aujourd'hui, il se présente en de très nombreuses variantes qu'on pourrait appeler des « dialectes ».

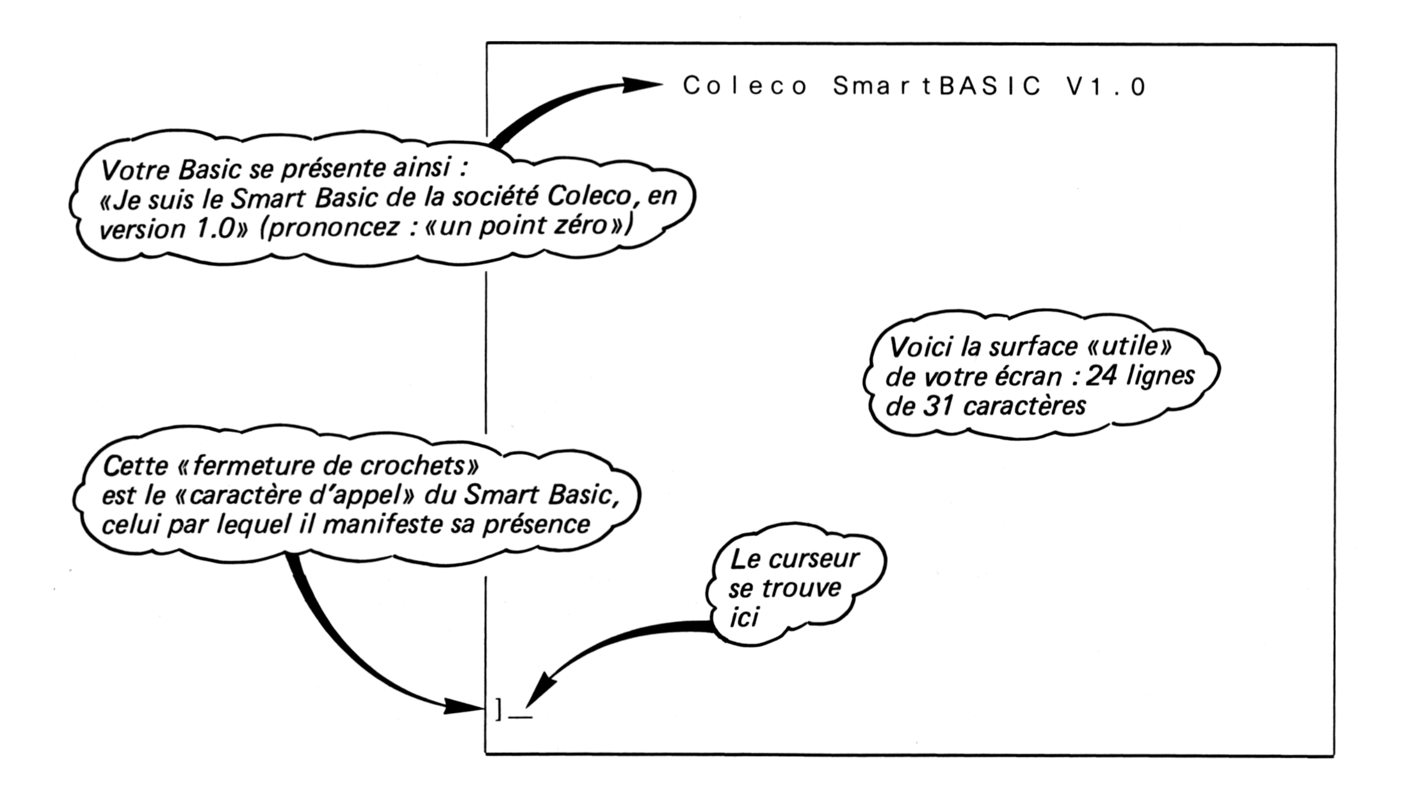
APPLESOFT

SMARTBASIC

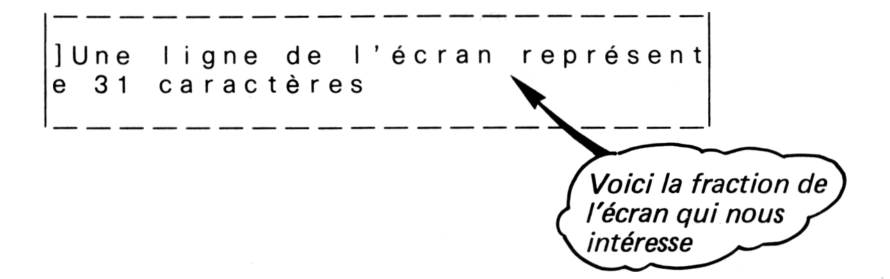
Pensez aux patois régionaux, tournant tous autour du français. Le Basic de votre Adam, lui, s'inspire très fortement de l'un des Basics les plus célèbres, « l'Applesoft » qu'utilise la société *Apple* pour ses micro-ordinateurs (Apple II et Apple III, par exemple). Pour Adam, ce Basic s'appelle *SmartBasic* (approximativement : *Basic élégant, efficace).*
En fait, le Basic est une sorte de traducteur qui accepte des ordres (anglais) et les traduit en informations compréhensibles par le microprocesseur (des signaux électriques en binaire). C'est *aussi* un programme : il est stocké dans l'une des cassettes qui vous sont livrées avec Adam. Repérez la cassette marquée SmartBasic et introduisez-la dans l'unité à cassettes, *l'ordinateur étant en marche.*

Il ne doit jamais se trouver de cassettes quelconques dans les unités à cassettes lorsque vous mettez l'ordinateur en service, ou hors service.

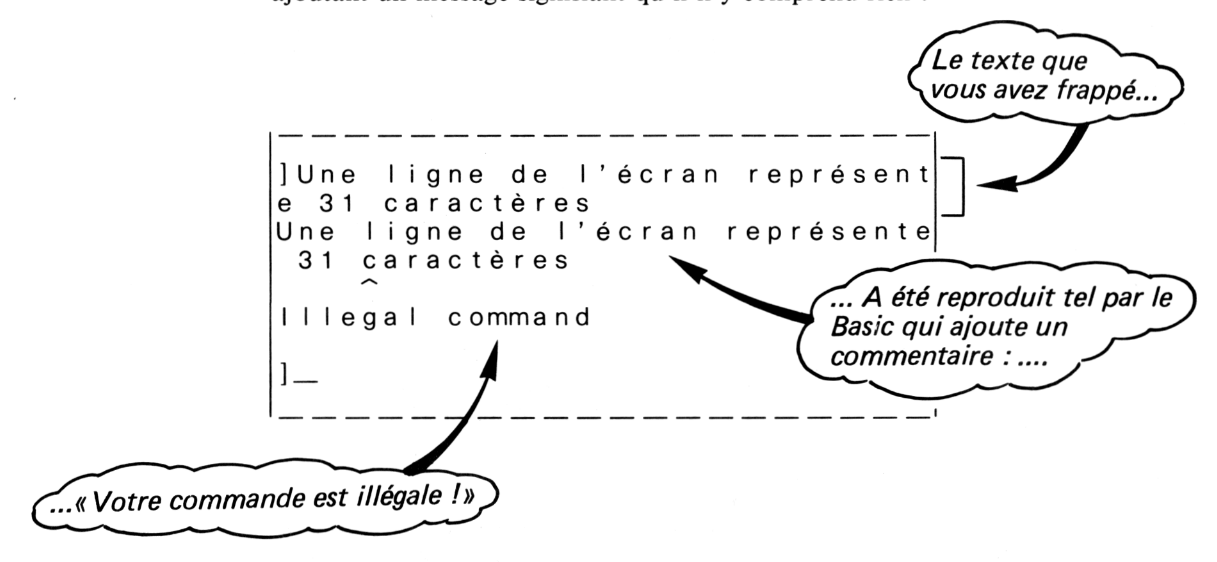
Puis, faites un RESET COMPUTER en actionnant le poussoir situé sur le dessus de l'unité centrale. Adam va tester l'unité à cassette et, découvrant qu'une cassette est présente, va lui accorder priorité et la lire ; il transfère son programme dans ses mémoires centrales, puis il en commencera l'exécution, ce qui donnera l'écran suivant après quelques instants de patience (nous ne vous montrerons désormais que la partie « active » de l'écran, ou encore des « coupures » de l'écran afin de ne vous exposer que ce qui est intéressant) :



Frappez donc le texte suivant, afin de vérifier la longueur d'une ligne affichée :



Cette fois, il ne s'agit plus de traitement de texte et les mots sont coupés en bout de ligne ! Si vous frappez sur la touche RETURN, le Basic va reproduire ce texte en ajoutant un message signifiant qu'il n'y comprend rien :



En effet, cette phrase ne répond à *aucun* des codes d'honneur du Basic (patience !). Puis, le « caractère d'appel du Basic est revenu deux lignes en dessous, suivi par le curseur clignotant.

MESSAGES D'ERREUR

Le Basic a la bonté d'émettre des messages chaque fois qu'il détecte que vous avez commis une erreur. Leur liste est longue !

Nous vous la donnons en fin d'ouvrage ; vous apprendrez vite à reconnaître les principaux d'entre eux.

Ces messages d'erreur, tout comme vos frappes, ne font en aucun cas souffrir l'ordinateur et ne peuvent lui nuire. Ils font partie de la vie courante.

Remarquez qu'au fur et à mesure de vos frappes, l'écran « se déroule » : le texte *monte* et les lignes supérieures se perdent.

FORMAT DE L'ECRAN

Au total, vous pouvez afficher 24 lignes de 31 caractères, éventuellement le caractère d'appel du Basic compris.

Passons à nos premiers exercices.

2. L'instruction PRINT

CONTROLE

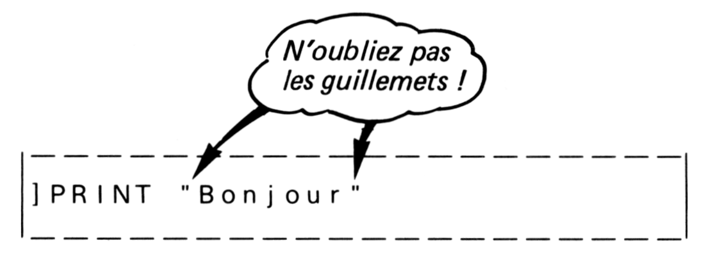
Commençons pas effacer l'écran, qui doit vous paraître très chargé. Pour cela, il faut employer la touche marquée CONTROLE *(commande),* qui fonctionne comme SHIFT : il faut la maintenir enfoncée puis frapper une autre touche simultanément, et enfin relâcher le tout.

EFFACEMENT DE L'ECRAN

CONTROLE/L

Pour effacer l'écran, la combinaison est CONTROLE puis la touche L, ce qu'on indiquera par CONTROLE/L.

Vérifiez-le : l'écran s'efface et le curseur clignotant se place tout en haut et à gauche. Nous vous donnons, en fin d'ouvrage, une liste de fonctions avec CONTROLE. Puis, frappez en respectant bien le code suivant (n'oubliez pas les guillemets) :

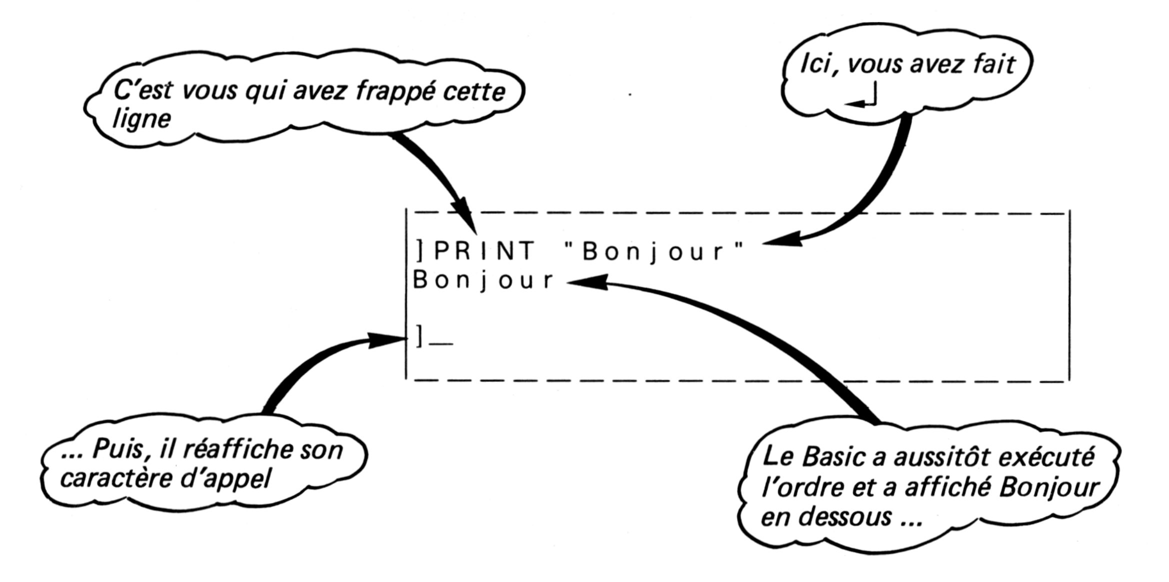


Le mot PRINT signifie, ici, *afficher sur l'écran* et non imprimer avec l'imprimante ; c'est un ordre d'affichage.

PRINT

Vous pouvez frapper des commandes telles que PRINT aussi bien en majuscules qu'en minuscules *(print),* peu importe. Le Basic les interprètera toujours comme si elles étaient en majuscules, aussi les écrira-t-on systématiquement ainsi.

Puis frappez sur le « retour chariot », la touche RETURN, notée aussi ↵. L'écran va vous montrer :

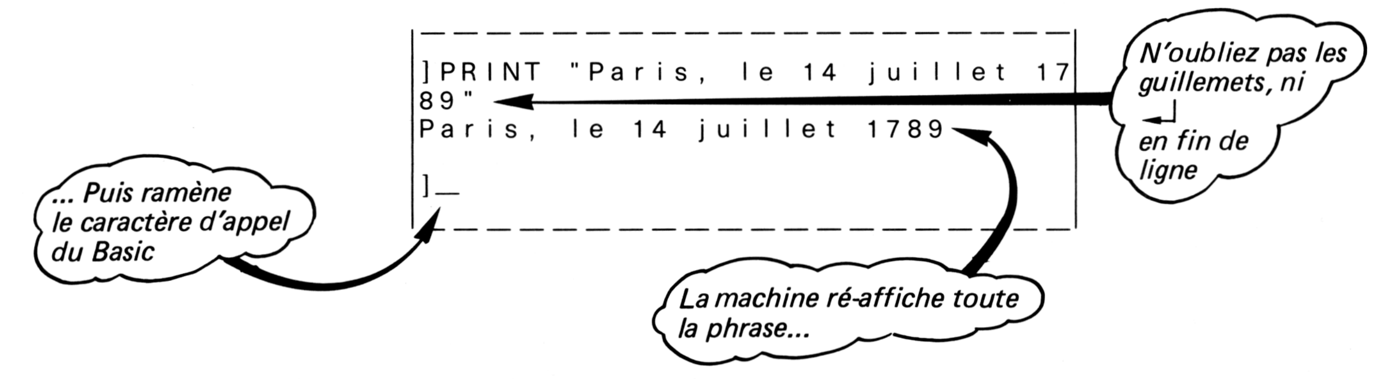


La première conclusion importante est la suivante :

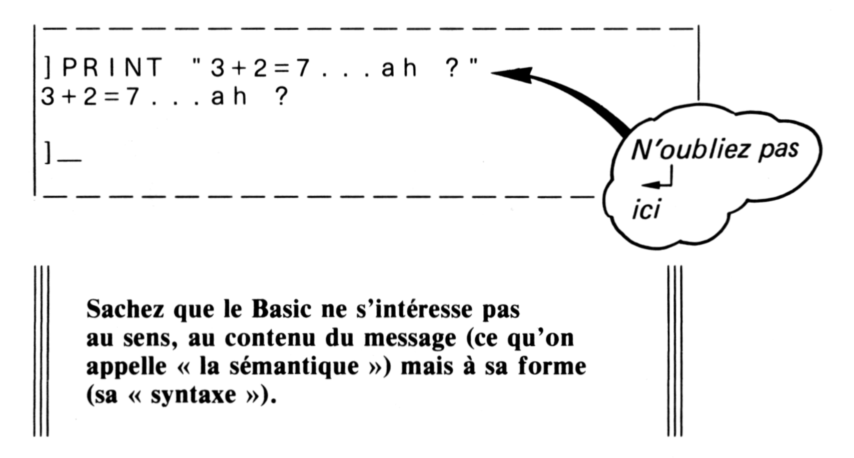
RETURN, OU ↵

L'ordre frappé n'est pris en compte que lorsque vous faites un RETURN (↵). C'est, en quelque sorte, une validation ; elle lance aussitôt l'exécution.

Faites une nouvelle expérience avec PRINT ; écrivez la phrase suivante, puis faites un RETURN (↵) :

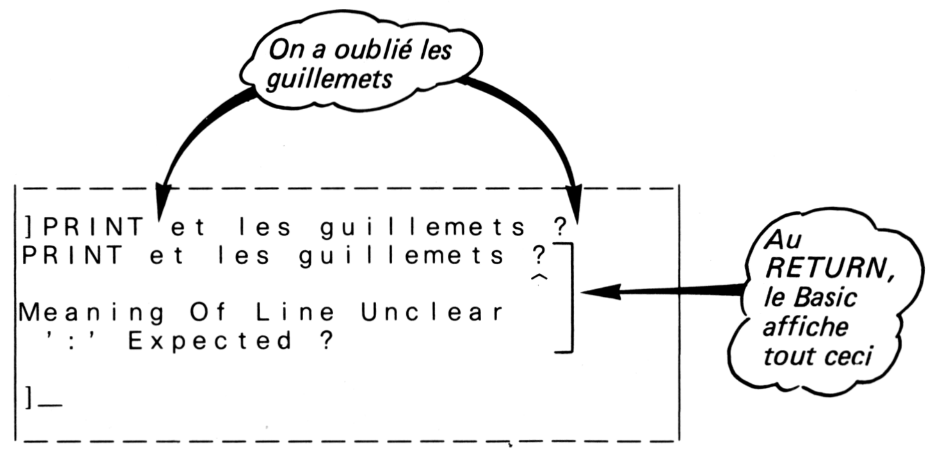


A nouveau, le Basic a fidèlement reproduit votre texte ; celui-ci peut être absolument quelconque, à la condition qu'il soit encadré par des guillemets, par exemple :



Sachez que le Basic ne s'intéresse pas au sens, au contenu du message (ce qu'on appelle « la sémantique ») mais à sa forme (sa « syntaxe »).

Ainsi, les guillemets lui sont indispensables. Si vous les omettez, rien n'ira plus :



GUILLEMETS

S'il manque les guillemets, le Basic ré-affiche toute la ligne, puis il pointe une tête de flèche (l'accent circonflexe) vers la ligne erronée ; parfois, celle-ci se positionnera sur l'erreur détectée. Enfin il affiche : « *La signification de cette ligne n'est pas claire. Fallait-il attendre un ':' ?* ». Voici une sortie imprimante de ce texte :

```
]print et les guillemets ?
print et les guillemets ?
                        ^
Meaning Of Line Unclear
 ':' Expected ?

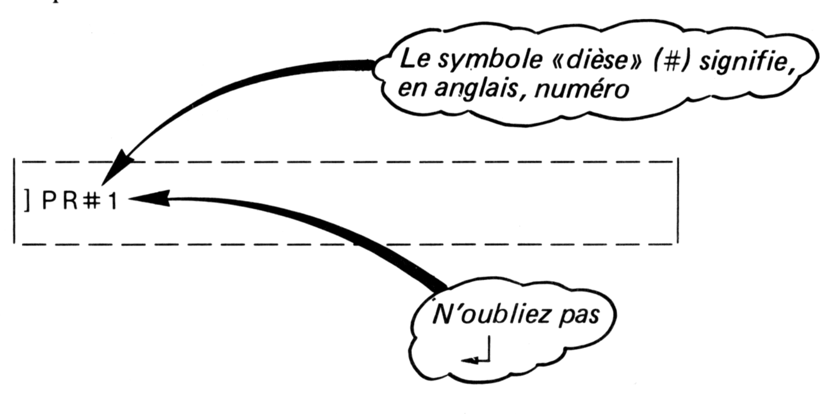
]
```

Nous utiliserons, par la suite, des sorties sur imprimante réelles ou simulées, afin d'illustrer nos explications. Peut-être voudriez-vous savoir aussitôt comment les obtenir ? Rien de plus simple :

IMPRESSION DE L'ECRAN
CONTROLE/P

• Pour reproduire sur papier l'écran *complet,* frappez simplement CONTROLE/P (maintenez CONTROLE enfoncée, frappez P, puis relâchez le tout). Cela suffit : l'imprimante s'active et « recopie » l'écran.

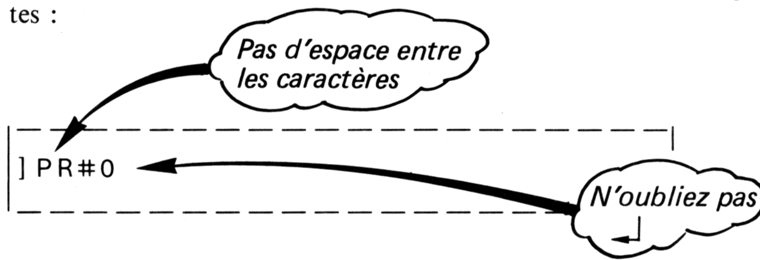
• Pour diriger vers l'imprimante le texte que vous frappez *ou* les messages, donnez au préalable l'ordre :



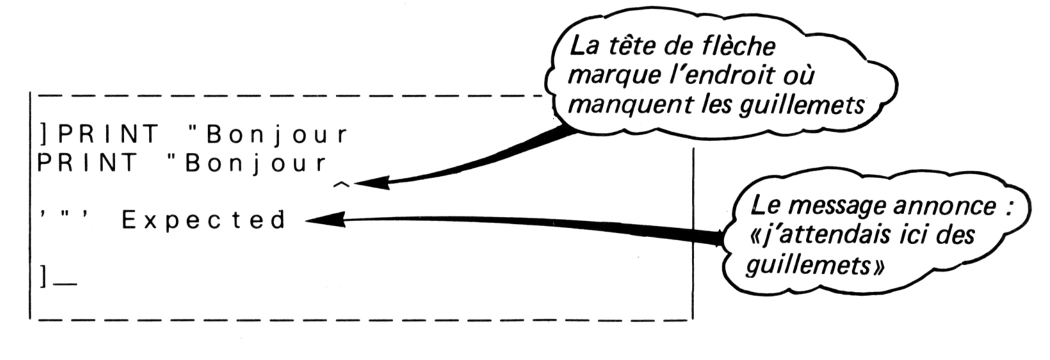
PR#1
PR#0

C'est le code « secret » qui dirige les affichages vers l'imprimante :
— soit après la frappe de RETURN (↵) si vous avez écrit une ligne ;
— soit les messages du Basic au fur et à mesure de leur affichage.

• Enfin, pour supprimer l'action de PR#1 et revenir à l'affichage uniquement, faites :



Tout d'abord, nous allons vous prouver que la tête de flèche peut effectivement marquer l'erreur ; ici, on a omis de *refermer* les guillemets : le message est explicite :



Notez que sur l'écran, la ligne que nous avons frappée est *doublée,* répétée dans sa totalité par le Basic, mais non imprimée. Cela a peu d'importance pour nos exemples.

Après chaque opération, le Basic ramène son caractère d'appel] et le curseur. Nous vous les avons indiqués sur nos « vues », jusqu'ici, mais nous allons maintenant nous en dispenser quand ce ne sera pas réellement nécessaire. Cela vous manquera-t-il ?

3. Corrections

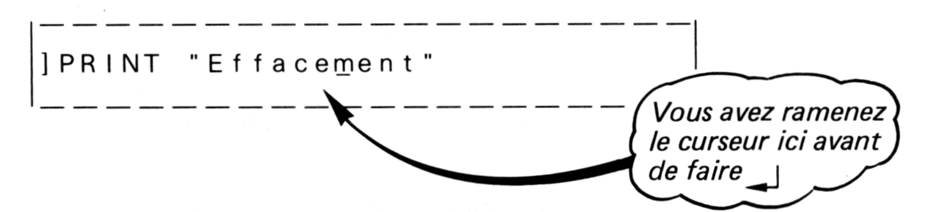
Bien sûr, vous allez commettre bien des erreurs de frappe avec nos exercices, aussi nous devons-nous de vous indiquer aussitôt comment les corriger.
Pour supprimer *un caractère précédent* (ou des caractères) avant d'avoir frappé le ↲, vous pouvez, au choix, utiliser :

- la touche d'effacement arrière marquée RETOUR/ESPACE ;
- la touche de gestion du curseur « flèche à gauche » (←).

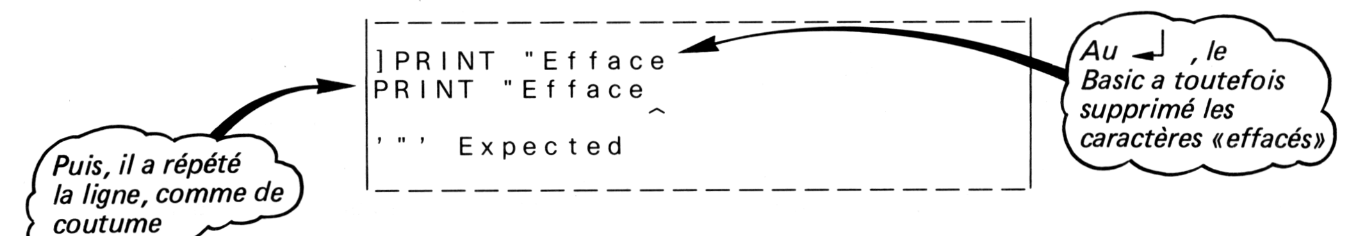
SUPPRESSION D'UN CARACTERE
RETOUR/ESPACE ET ←

Notez que leur action est, ici, semblable : elles emmèneront le curseur sous les caractères précédents. Ceux-ci ne sont pas effacés de l'écran mais disparaissent bel et bien de la mémoire de l'ordinateur.

Vérifions-le en frappant la ligne suivante, puis en *ramenant* le curseur sous le *m* avant de faire RETURN (↲) :



Si nous ne vous avons pas menti, les lettres « ment » ont disparu. Prouvons que c'est exact en faisant maintenant RETURN ; on obtient, avec l'inévitable message d'erreur :



Tant que les caractères supprimés sont encore apparents sur l'écran, vous pouvez les « ré-installer » sans les refrapper, simplement en promenant le curseur *vers la droite,* sous eux, à l'aide de la touche *flèche à droite* uniquement.

RECOPIE DE CARACTERES

Tous les caractères que croise le curseur sont « recopiés » en mémoire de la machine, mais ce, uniquement avec la touche « flèche à droite ».

Vérifiez-le en reprenant le PRINT « Effacement », en ramenant le curseur sous le *m,* puis en le repropulsant vers la droite avec →, après la fermeture des guillemets. Au ↵, le mot *Effacement* se ré-inscrit au complet.

EFFACEMENT D'UN CARACTERE SUR L'ECRAN

CONTROLE/O

Si vous teniez absolument à effacer un caractère sur l'écran, frappez CONTROLE/O (maintenez CONTROLE enfoncée, frappez sur la lettre O puis relâchez le tout) : le caractère marqué par le curseur disparaît.

ANNULATION D'UNE LIGNE

CONTROLE/X

Si vous voulez annuler une ligne complète *avant* d'avoir fait RETURN, frappez l'ordre CONTROLE/X (maintenez CONTROLE enfoncée, frappez sur la lettre X et relâchez le tout). Le Basic va placer une barre oblique *inversée* en fin de ligne, ignorera totalement celle-ci, et ramènera le curseur au début de la ligne suivante :

La frappe de CONTROLE/X se traduit par ce symbole ; cette ligne est annulée

```
]Supprimons cette ligne\
```

Vous en savez assez sur ces suppressions, pour l'instant.

4. Premiers calculs

Le mot PRINT demande 5 frappes ; si vous voulez « vous économiser », remplacez-le par une *unique frappe,* celle d'un point d'interrogation (?).

? POUR PRINT

Pour le Basic, le point d'interrogation en début de ligne a le sens de PRINT et le remplace.

Essayez avec un PRINT « Bonjour » :

```
]? "Bonjour"
Bonjour
```

Nous allons systématiquement employer ce ? à la place de PRINT ; ne vous y trompez pas. Faites maintenant l'expérience suivante ; frappez d'abord :

```
]? "3+2"
3+2
```

Maintenant, reprenez la même formule mais *sans* les guillemets :

```
]?  3+2
5
```

Surprise ! Non seulement il n'y a pas d'erreur mais encore, le Basic a exécuté le clacul.

OPERATIONS SUR LES CHIFFRES

Les chiffres n'imposent pas l'emploi des guillemets :

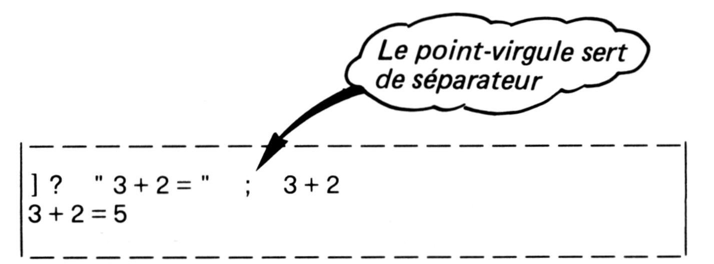
• avec des guillemets, ils sont reproduits tels ;
• sans guillemets, le Basic exécute l'opération s'il s'en trouve une. Sinon, il recopie ces chiffres.

Par exemple, il n'y a pas d'opération ici ; par conséquent, les chiffres sont reproduits tels :

```
]?  12345
12345
```

Combinons les deux modes, *avec* et *sans* guillemets, en séparant nos deux formules avec un point-virgule, pour voir ce que cela donne :

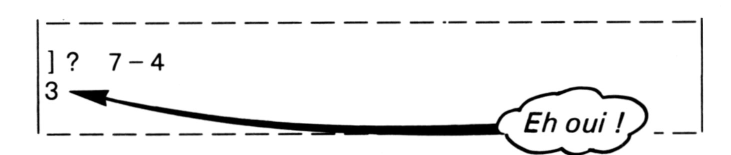
POINT-VIRGULE DANS UN PRINT



La première expression entre guillemets a été reproduite telle, alors que la seconde a donné lieu à exécution, d'où le résultat.

Le point-virgule sert de séparateur « officiel » dans un ordre PRINT. Il évite d'avoir à répéter cet ordre. Notez aussi et surtout qu'il commande les affichages successifs *sur la même ligne*.

Essayons une soustraction simple :

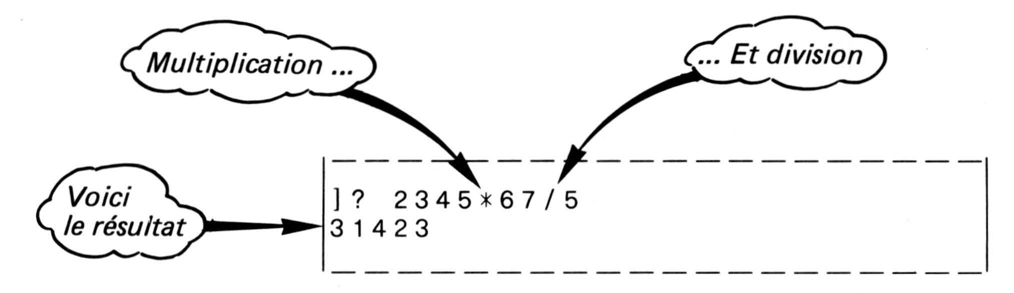


Passons à plus difficile, la multiplication et la division, à propos desquelles il faut noter deux points importants :

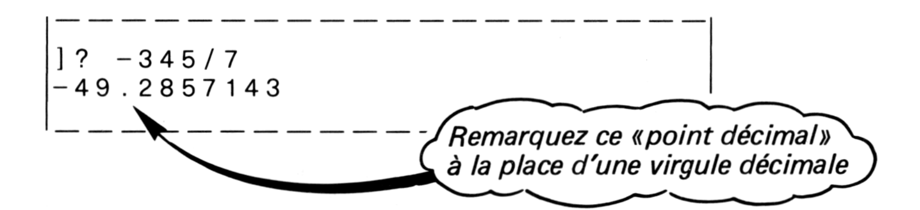
MULTIPLICATION (*)
DIVISION (/)

En informatique, le symbole de la multiplication est l'astérisque (*) et celui de la division, la barre oblique (/).

Ainsi, on ne risque plus de confondre le signe de la multiplication avec la lettre X comme c'est le cas en arithmétique, lorsqu'on emploie le symbole courant *multiplié par* (×). Posons une opération compliquée avec multiplication et division :



Passons maintenant à une opération donnant à la fois un résultat négatif et une valeur décimale :



Admirez la précision du résultat, fourni instantanément avec 7 décimales. Nous devons aussi attirer votre attention sur le fait que le Basic, d'origine américaine, place un *point décimal* là ou nous, nous mettons d'habitude une *virgule décimale.* En effet, nous écrirons le nombre « pi » 3,14 mais le Basic le notera 3.14.

POINT DECIMAL
VIRGULE DECIMALE

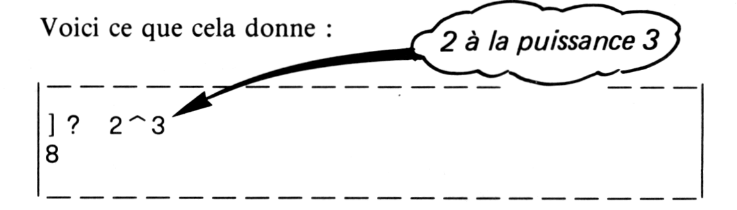
N'oubliez pas que le point décimal remplace la virgule décimale. Ne vous trompez pas, en écrivant des nombres car le Basic ne les comprendrait pas.

Désormais, nous écrirons tous les nombres décimaux avec un point. Mais maintenant que vous connaissez la règle, entraînez-vous à exécuter toutes sortes d'opérations. Vous apprécierez ce « mode direct » qui fait de votre ordinateur une... bonne calculette. Mais puisque nous en sommes à l'arithmétique, voyons comment élever un nombre à une puissance, par exemple 2 puissance 3, ce qui s'écrit en arithmétique 2^3 : mais pas en Basic.

PUISSANCES

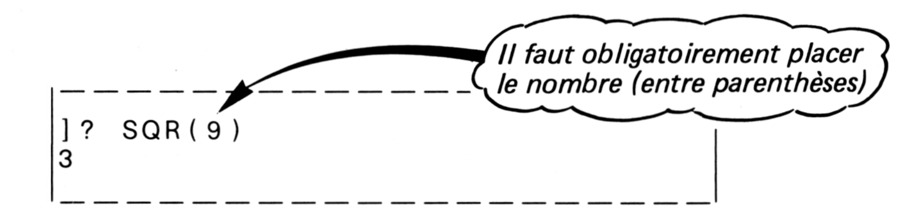
Le symbole de l'élévation à puissance est la tête de flèche verticale, qui est une sorte d'accent circonflexe que l'écran placera *sur* la ligne.

Voici ce que cela donne :



L'extraction d'une racine carrée, elle, ne fait pas appel à un symbole mais au code SQR, lequel provient du mot anglais « *square* », carré :

RACINE CARREE
SQR



Voilà : exercez-vous !

5. Chaînes de caractères

On a vu que des chiffres ou des opérations numériques pouvaient être dispensés de guillemets. Par exemple, on peut écrire PRINT 123 ou des choses plus compliquées sans guillemets.

ENSEMBLES NUMERIQUES

Dans ce cas, ces chiffres ou ces opérations constituent ce qu'on appelle en informatique des ensembles numériques, mais aussi parfois des chaînes numériques.

Ainsi, le seul chiffre 1 est un ensemble numérique (de un seul caractère), mais 123 *456 *sans guillemets* est aussi un ensemble numérique.
Or, on oppose à ces *ensembles numériques* tout ce qu'on encadre de guillemets, ainsi qu'on l'a fait au début de ce chapitre.

CHAINE DE CARACTERES

Tout ce qui est encadré de guillemets devient, en informatique, une « chaîne de caractères ».

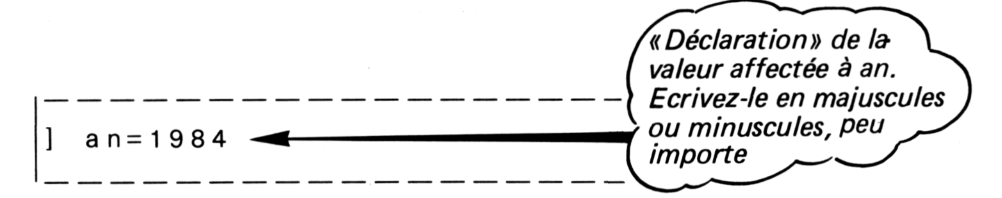
Ainsi, « Bonjour » est une chaîne de caractères, mais aussi « 2 + 3 » ou encore, en mélangeant lettres et chiffres, « Paris, le 14 juillet 1789 ». Sachez qu'on appelle *aussi* « chaîne de caractères » :

- un simple espace introduit entre des guillemets : " "
- ou même *rien du tout* dans ces guillemets : "".

Ces notions d'*ensembles numériques* (encore appelées *valeurs numériques)* qui ne peuvent comprendre que des chiffres ou des opérations *sans* guillemets, et de « chaînes de caractères » dans lesquelles on peut tout mêler pourvu qu'on encadre ce tout de guillemets est très importante en Basic, on va le voir.

6. Variables et constantes

Avez-vous la mémoire des chiffres ? Si non, vous serez heureux d'apprendre que le Basic accepte qu'on les remplace par un nom, un peu comme en arithmétique, d'ailleurs, où l'on écrit des choses du genre de X = 3. Par exemple, on va décider de remplacer l'année 1984 par le nom AN, sur deux lettres ; utilisez majuscules ou minuscules, peu importe au Basic qui confondra les deux.
Pour commencer, il faut annoncer à haute et intelligible voix que, désormais, le mot AN remplacera 1984 et aura cette signification. Pour cela, remplissez la déclaration de la façon suivante :



Désormais, le Basic a rangé cela dans ses mémoires. En voulez-vous la preuve ? Demandez-lui d'afficher an : il vous restituera le nombre qui lui a été affecté :

```
] ? an
1984
```

Ou encore en majuscules (ce qui revient au même) :

```
] ? AN
1984
```

Cela fonctionne parfaitement avec les chaînes numériques, et tout aussi bien avec les « chaînes de caractères ». Mais là, il faudra spécifier au Basic qu'il s'agit d'une « chaîne de caractères » en ajoutant un code spécial au *nom* attribué à cette chaîne.

SYMBOLE " DOLLARS "

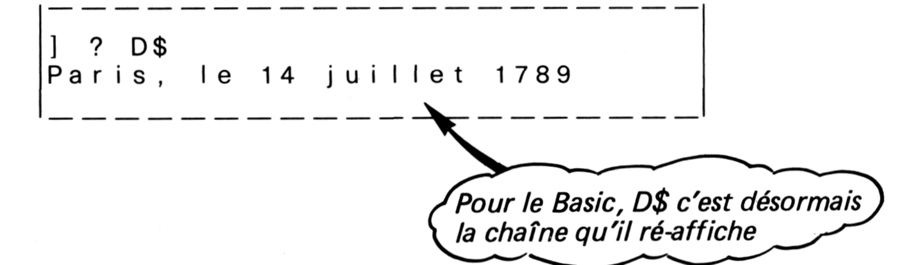
Pour marquer qu'il s'agit d'une « chaîne de caractères », on ajoute en suffixe le symbole *dollars*, qui s'écrit $, au nom attribué.

Prenons un exemple. On va appeler D$ l'en-tête de lettre « Paris, le 14 juillet 1789 », avec D pour... *date* (trouvez mieux !) et le suffixe *dollars* pour marquer qu'il s'agit d'une chaîne de caractères. « Déclarons-le » au Basic :

Remarquez que depuis peu, on introduit un espace en début de ligne. Ce n'est pas indispensable mais ne trouvez-vous pas que c'est plus joli ?

```
] D$="Paris, le 14 juillet 1789
```

Cela fait, demandons au Basic de nous rappeler la date, avec un PRINT D$:

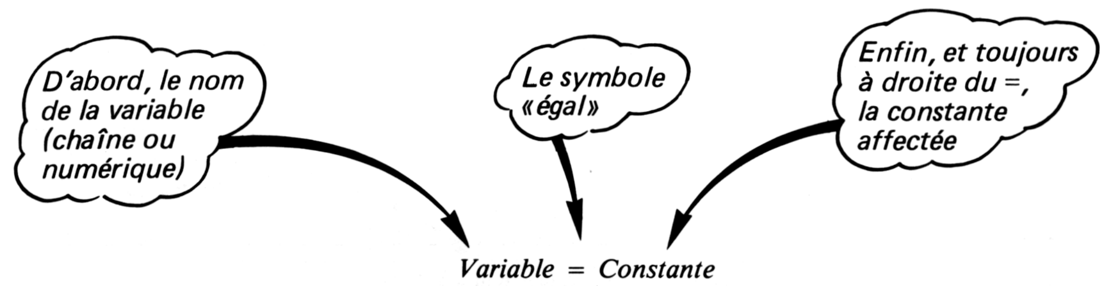


Généralisons cela, maintenant : le *nom* attribué à la chaîne (numérique ou de caractères) s'appelle une *variable* dans ce langage propre à l'informatique. La chaîne (numérique ou de caractères) affectée à cette variable s'appelle une *constante.* Ainsi, AN est un nom de *variable numérique,* alors que D$ est un nom de *variable chaîne ;* en outre, 1984 est une *constante numérique* alors que « Paris, le 14 juillet 1789 » est une *constante chaîne.*

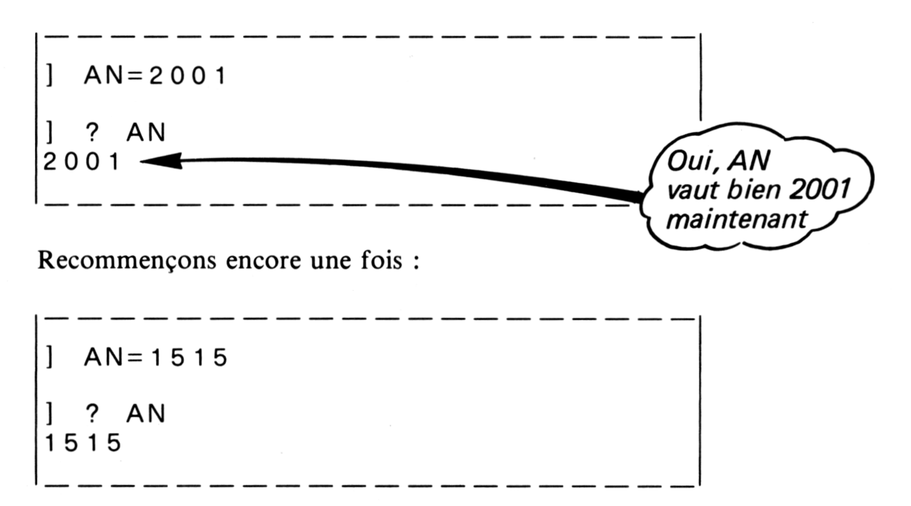
VARIABLE
CONSTANTE

Rappelez-vous bien que la variable est un nom général, la constante est la chaîne qui lui est affectée.

Lorsque nous avons « déclaré » les affectations au Basic, nous avons utilisé cette formule obligatoire :



Cela signifie que vous n'avez pas le droit d'écrire l'inverse, par exemple 1984 = AN. Rappelez-vous aussi que dans le cas de chaînes de caractères, la variable est suffixée par le symbole $.
Pourquoi *variable* et *constante ?* Tout simplement parce que la *variable* peut représenter des *constantes* successives ; vérifions-le en affectant une nouvelle valeur à AN, 2001, par exemple : elle va prendre la place de l'ancienne qui est perdue :



```
] AN=2001

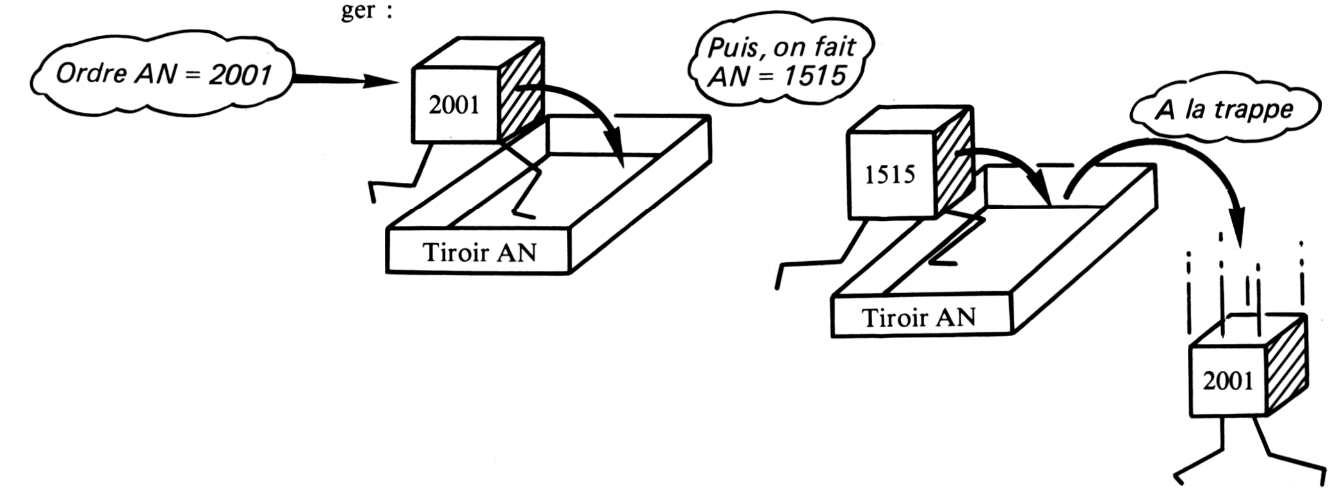
] ? AN
2001
```


Recommençons encore une fois :

```
] AN=1515

] ? AN
1515
```

On peut dessiner ainsi ce phénomène : la variable représente une boîte qui porte le nom de la variable. La constante est ce que l'on met dans la boîte et qui peut changer :

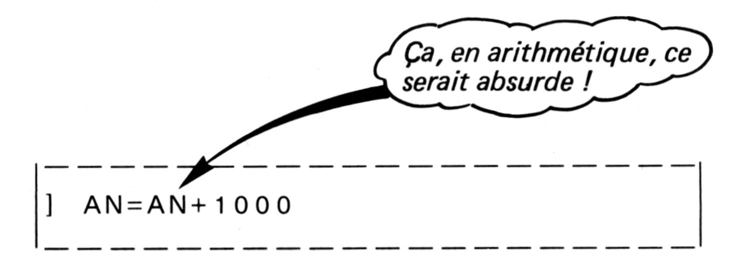


En réalité, il ne s'agit pas d'une boîte mais d'une cellule de la mémoire vive, centrale, de l'ordinateur, à laquelle on décerne un nom : le Basic comprendra qu'il doit la réserver. Puis, on pourra y loger des constantes successives à volonté.
Tout cela paraît simple... mais le symbole d'affectation = utilisé constitue un piège ; les informaticiens ont recherché un symbole pour marquer ce type d'*affectation* parmi ceux disponibles sur un clavier courant. A défaut d'autre chose, ils ont adopté le signe *égal.*

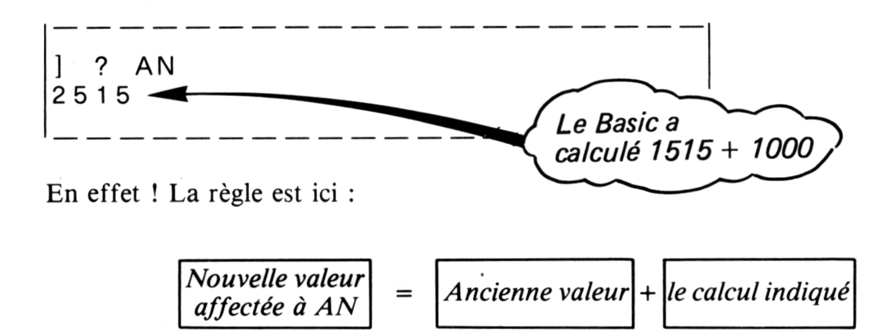
SYMBOLE D'AFFECTATION =

> **Attention : ce symbole « égal » ne possède plus, ici, le sens d'une égalité mathématique. Il veut dire uniquement : « se voit affecter ».**

Ainsi, AN = 1515 signifie que *AN se voit affecter 1515.* Pour bien comprendre ce point, faites maintenant l'expérience suivante en utilisant une formule de *déclaration d'affectation* qui serait absurde du point de vue arithmétique :

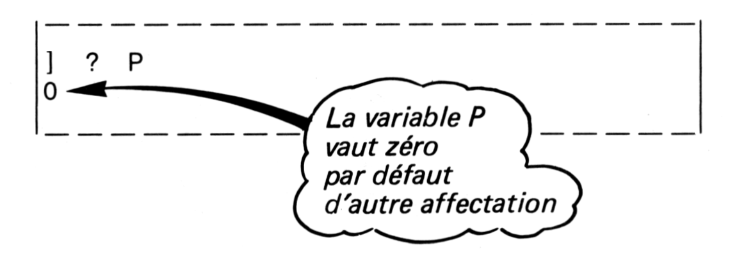


Qu'est-ce que cela va-t-il donner ? Le Basic, tout simplement, a calculé l'expression *à la droite* du symbole = ; il a pris 1515 pour AN, lui a ajouté 1000 ce qui donne 2515, et a rangé cette nouvelle valeur dans la boîte AN. Vérifions-le :

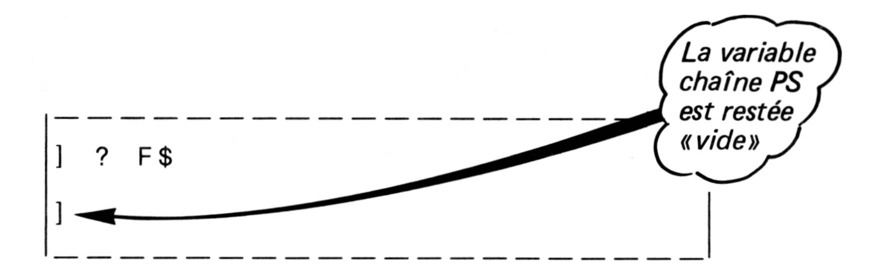


En effet ! La règle est ici :

On vous le disait que = n'est plus un symbole d'égalité, mais d'affectation. C'est toujours déroutant pour le débutant.
Terminons ceci en tendant un piège à l'ordinateur. Demandons-lui combien vaut une variable numérique à laquelle on n'a encore affecté *aucune* constante, par exemple la variable P :



Par principe, le Basic attribue un zéro à toute variable à laquelle on n'a pas pas affecté de constante ; cela, pour les numériques. Car pour les variables chaînes, c'est encore pire : il ne daigne même plus leur attribuer un zéro, les « boîtes » sont désespérément vides, on ne lira rien ; ainsi, avec F$ à qui l'on n'a rien affecté :



Sachez encore que, pour le Basic, P et P$ sont *deux* noms de variables différents, l'un, P, est un nom de variable numérique et l'autre, P$, de variable chaîne.

7. Les noms des variables

Pour terminer, voici les règles à respecter pour nommer les variables :

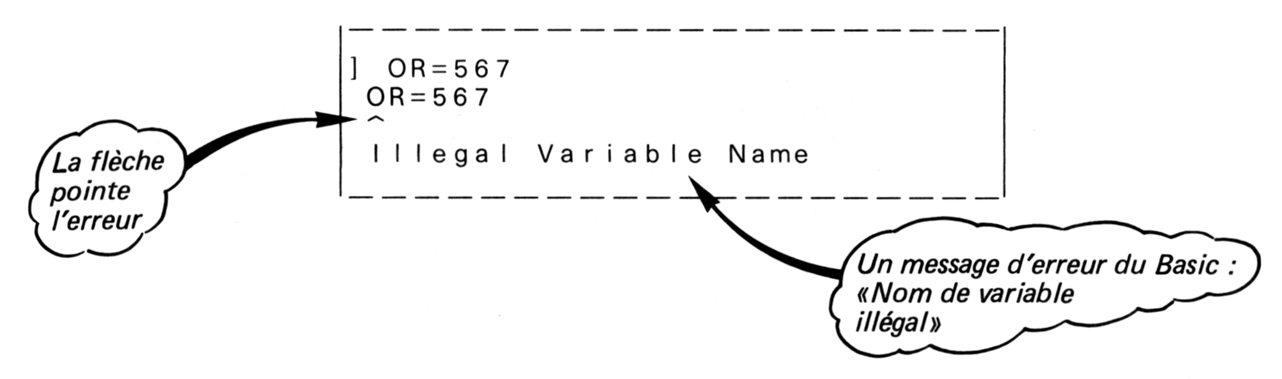
- le nom d'une variable peut être une lettre,
- il peut être composé de deux lettres,
- ou encore, il peut comprendre une lettre et un chiffre,
- à ce nom et pour les variables « chaînes », on ajoutera le suffixe $.

NOMS DES VARIABLES

Toutes les autres formules sont interdites ou à fuir ! Par conséquent, P1 est *autorisé* mais l'inverse 1P est *interdit :* d'abord la lettre !

MOTS RESERVES

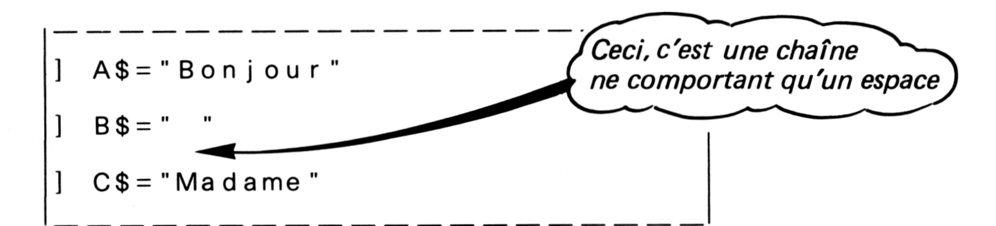
Mais ce n'est pas tout : le Basic utilise des mots qui lui sont propres et dont il se réserve l'emploi. Il vous est interdit de les reprendre comme noms de variables. Ces mots sont appelés des *mots réservés ;* ils comprennent, par exemple, des mots de *deux* lettres tels que ON, ou OR qu'il vous est donc interdit d'employer pour autre chose. Voici ce qui vous arriverait si vous essayiez :



Par contre, ils ne comprennent pas de nom d'une seule lettre, ou de noms d'une lettre avec un chiffre.

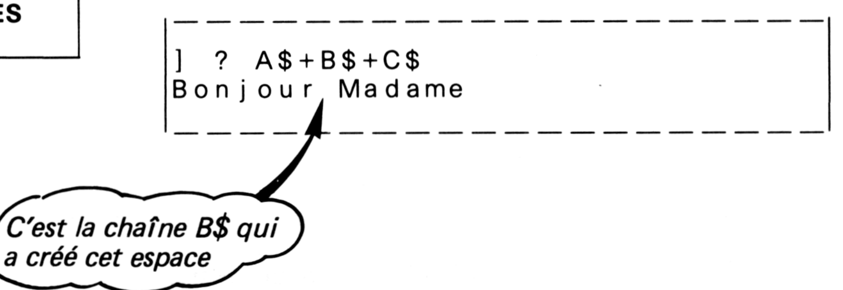
Un conseil : pour vos noms de variables, n'utilisez alors que des noms d'une seule lettre, ou encore d'une lettre suivie d'un chiffre.

Vous ne risquerez donc rien en procédant ainsi, mais essayez de trouver des noms pas trop abstraits, par exemple M1$ pour le *mois* de « Janvier », M2$ pour « Février », etc.
Pour en finir avec ce chapitre, voyons comment on peut mettre des chaînes de caractères bout à bout, les « concaténer », comme l'on dit. Par exemple, déclarons les chaînes suivantes :



CONCATENATION DE CHAINES DE CARACTERES

Si, maintenant, vous voulez *concaténer* ces chaînes, il vous faudra utiliser le symbole + lequel, à nouveau, ne possède pas son sens arithmétique habituel mais signifie « mettre bout à bout » :



C'est là-dessus que nous terminerons ce chapitre, consacré au *mode direct :* vous frappez un ordre et vous frappez sur RETURN — ce que l'on désigne par : *entrer un ordre,* ou *introduire un ordre en machine —,* et il est *aussitôt* exécuté.

Notez, enfin, qu'à chaque fois que vous remettez l'ordinateur en service et voudrez travailler en Basic, il vous faudra « le recharger » à partir de la cassette, ainsi qu'on l'a vu.

Désormais, et dès que le Basic est rechargé, prenez l'habitude de retirer la cassette SmartBasic et de la ranger. Sachez enfin que ce Basic va être logé dans les mémoires vives d'Adam et occupera une place non négligeable : sur les 80 K octets disponibles à l'origine, il ne vous restera plus que 25 950 octets. Cela, en Basic, car en traitement de texte, les 80 K restent à votre disposition.

Questions sur le chapitre III

A nouveau, cochez la bonne réponse.

1. En mode Basic, votre écran peut afficher jusqu'à :
   - ☐ 24 lignes de 31 caractères
   - ☐ 32 lignes de 80 caractères

2. Pour effacer l'écran, vous frapperez :
   - ☐ Sur la touche EFFACER
   - ☐ La combinaison CONTROLE/L

3. PRINT signifie, en Basic :
   - ☐ Afficher sur l'écran
   - ☐ Imprimer avec l'imprimante

4. Que donnera l'ordre : PRINT 2 + 3 après le RETURN (ou ↵) :
   - ☐ 2 + 3
   - ☐ 5

5. Et celui-ci : PRINT "2 + 3"
   - ☐ 2 + 3
   - ☐ 5

6. Et celui-ci : PRINT BONJOUR
   - ☐ BONJOUR
   - ☐ Un message d'erreur

7. Et celui-ci : PRINT "3 × 7 = ";3 × 7
   - ☐ 3 × 7 = 21
   - ☐ 3 × 7 = 30

8. N est un nom de variable :
   - ☐ Numérique
   - ☐ De chaîne de caractères

9. P$ est un nom de variable :
   - ☐ Numérique
   - ☐ De chaîne de caractères

10. Ecrire A = A + 1 en Basic :
    - ☐ Est une expression corecte
    - ☐ N'est pas correct

Réponses page 56

Réponses aux questions sur le chapitre III

1. En mode Basic, l'écran affiche jusqu'à 24 lignes de 31 caractères.

2. Pour effacer l'écran, il faut frapper la combinaison CONTROLE/L.

3. PRINT signifie « afficher sur l'écran », en Basic.

4. PRINT 2+3 donnera 5 (il n'y a *pas* de guillemets, donc le Basic exécute l'opération).

5. PRINT "2+3" donnera 2+3 en raison des guillemets.

6. PRINT BONJOUR, sans guillemets (oh, honte !) donnera une erreur.

7. Non ! Le symbole de la multiplication n'est pas le × mais l'astérisque (*). Le Basic n'y comprend plus rien et affiche 3×7=30, ce qui lui paraît évident (à lui !) : il a pris ×7 pour un nom de variable et lui a affecté un zéro d'office.

8. N est un nom de variable numérique.

9. P$ est un nom de variable chaîne de caractères.

10. Ecrire A=A+1 est parfaitement correct en Basic. Rappelez-vous que le symbole *égal* ne signifie pas une *égalité* arithmétique mais une *affectation*.

Comment vous en tirez-vous, ici ? Même si vos résultats ne sont guère brillants, ne vous en inquiétez pas trop car on va retravailler toutes ces notions, de façon pratique. Toutefois, n'hésitez pas à vous reporter à ce chapitre fondamental.

CHAPITRE IV

LE MODE PROGRAMMÉ

Le chapitre précédent ne constituait qu'une petite mise en condition : on va, maintenant, passer à la création de programmes réels. Pour cela, on va rédiger des listes d'instructions qui ne seront, cette fois, exécutées en salve que sur un ordre donné, et non plus une par une, au fur et à mesure de leur frappe. Ce mode s'appelle le « mode programmé », par opposition au « mode direct ».

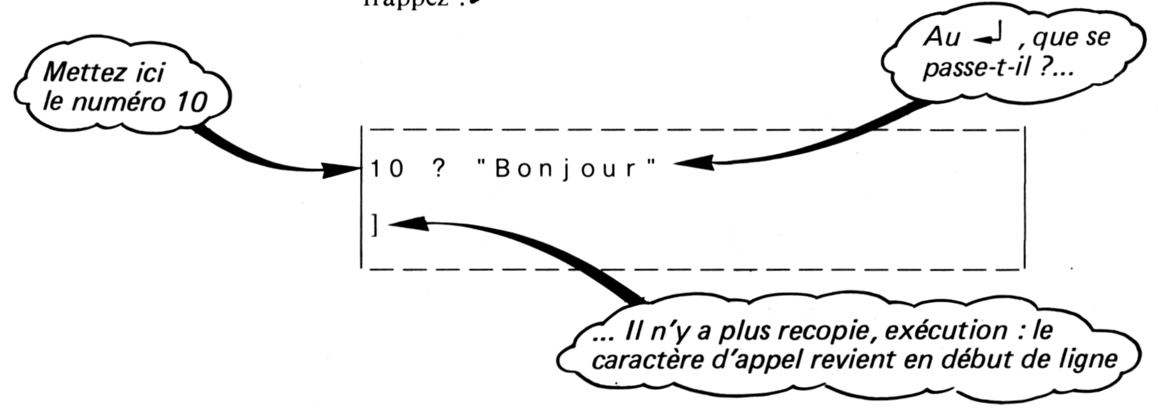
Principaux thèmes étudiés

Numérotation des lignes	Organigramme
RUN	REENTER
Adjonction d'une ligne	CONTROLE/C
LIST	Branchement inconditionnel
Mode programmé	Effacement programmé de l'écran
Remplacement d'une ligne	HOME
Suppression d'une ligne	Pour sauter une ligne
Messages d'erreur	CONT
Effacement de la mémoire	Code ASCII
NEW	CHR$
Programme de conversion	Listage partiel
INPUT	Modification d'une ligne
GOTO	Recopie d'une ligne

1. Numérotation des lignes

NUMEROTATION DES LIGNES

Dans ce mode *direct* étudié dans le chapitre précédent, un ordre frappé au clavier était immédiatement exécuté, dès la frappe sur ↵. Or, un programme informatique peut se composer de plusieurs ordres successifs (jusqu'à des milliers...) : par conséquent, il faut pouvoir les ranger en mémoire *sans* les exécuter au fur et à mesure ; on passera à cette exécution au seul moment opportun. C'est un tel mode qu'on appelle le *mode programmé,* ou encore le *mode programme.* Première expérience, frappez :



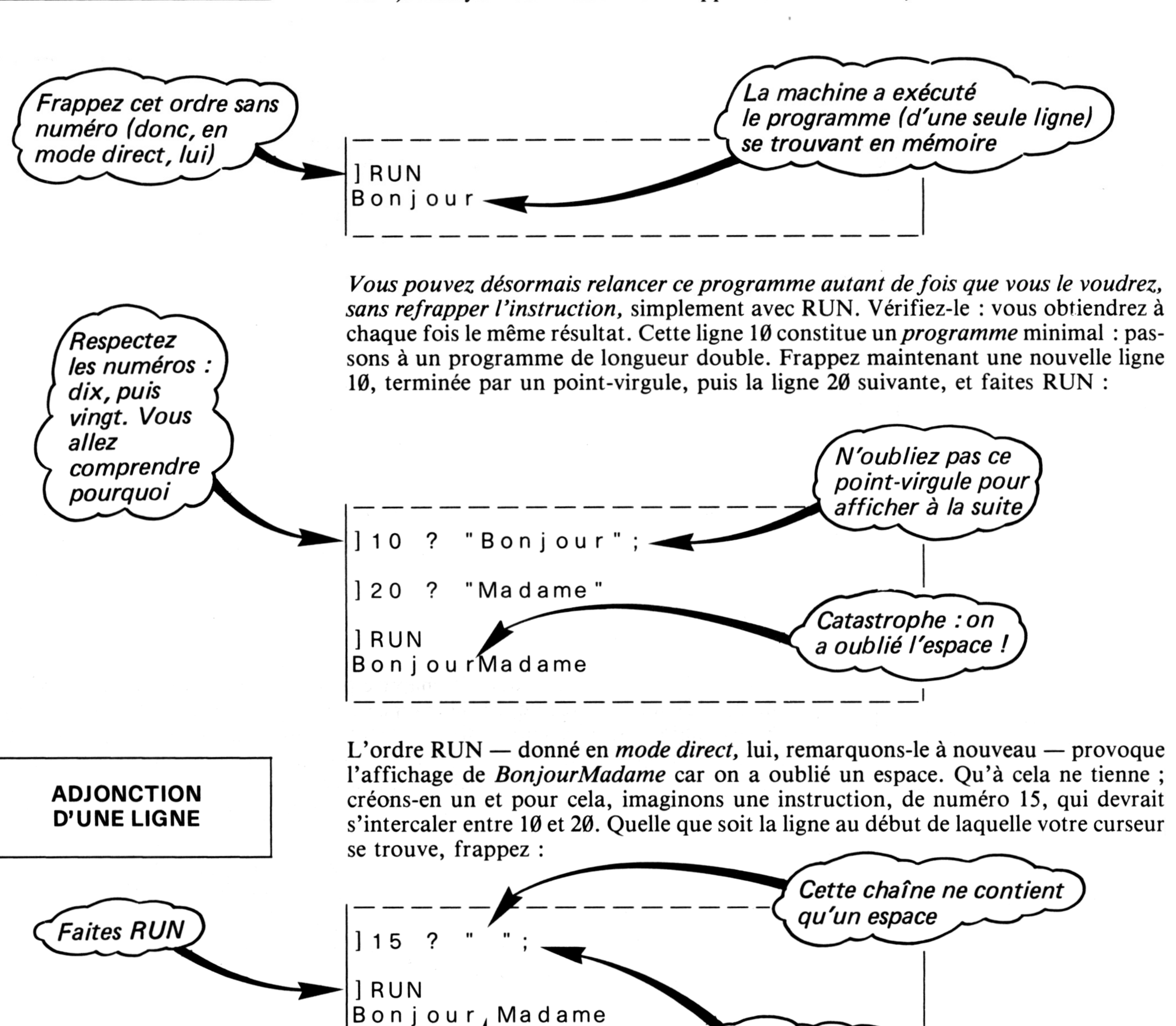
La machine n'a pas exécuté le PRINT « Bonjour » et le curseur est revenu à la ligne, attendant une nouvelle instruction.

> **Le seul fait de faire précéder l'instruction par un numéro d'ordre fait entrer la machine en « mode programmé ».**

Le numéro d'ordre, ou numéro de ligne d'instruction, peut être quelconque. Mettez 1, ou 2, ou 3, ou 78, ou 357,... peu importe : l'essentiel, c'est la présence d'un numéro.

RUN

Qu'est devenue l'instruction numérotée 1Ø, alors ? *Elle a été rangée en mémoire par le Basic, dès la frappe de* ↵. En voulez-vous la preuve ? On va lancer son exécution : pour cela, il faut utiliser le mot magique RUN (quelque chose comme : *Marche !*). Essayez tout d'abord de frapper ces trois lettres, suivies d'un RETURN :



Vous pouvez désormais relancer ce programme autant de fois que vous le voudrez, sans refrapper l'instruction, simplement avec RUN. Vérifiez-le : vous obtiendrez à chaque fois le même résultat. Cette ligne 1Ø constitue un *programme* minimal : passons à un programme de longueur double. Frappez maintenant une nouvelle ligne 1Ø, terminée par un point-virgule, puis la ligne 2Ø suivante, et faites RUN :

ADJONCTION D'UNE LIGNE

L'ordre RUN — donné en *mode direct,* lui, remarquons-le à nouveau — provoque l'affichage de *BonjourMadame* car on a oublié un espace. Qu'à cela ne tienne ; créons-en un et pour cela, imaginons une instruction, de numéro 15, qui devrait s'intercaler entre 1Ø et 2Ø. Quelle que soit la ligne au début de laquelle votre curseur se trouve, frappez :

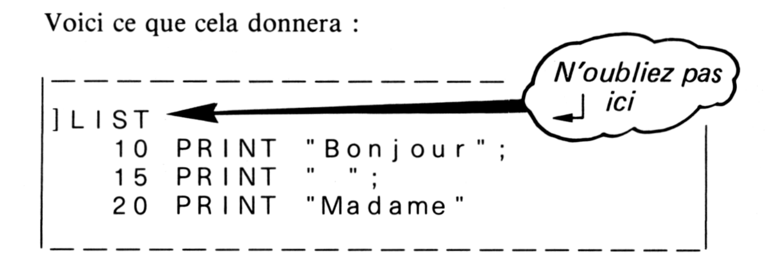
Avez-vous bien saisi, maintenant, le rôle du point-virgule dans un ordre PRINT ? Il supprime le retour à la ligne ; l'affichage suivant se fera alors sur la même ligne.

Pour comprendre ce qui s'est passé avec les lignes du programme, demandez au Basic de vous fournir le *listage* complet du programme qu'il a actuellement en mémoire, donc des trois lignes d'instructions que vous avez frappées.

LIST

Demandez le listage se dit LIST. Frappez ces quatre lettres, puis faites ↲. C'est donc un ordre en mode direct.

Voici ce que cela donnera :



Remarquez que le Basic a remplacé les points d'interrogation par le mot qu'ils représentent, PRINT. Il vous affiche vos trois lignes et a *introduit la ligne 15 entre la 10 et la 20.*

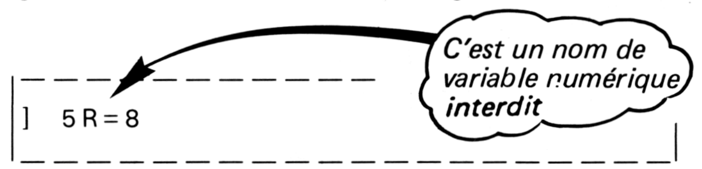
Vous comprenez, maintenant, pourquoi on a numéroté les lignes 10, puis 20 (on aurait continué par 30, puis 40, etc.) ? Lorsqu'on programme, on risque toujours d'oublier une instruction : elle sera facile à introduire si l'on s'est réservé des places dans la numérotation.

Tenez compte d'un autre phénomène important : le Basic ne s'intéresse qu'à l'ordre croissant des numéros, et non à leur valeur « absolue ». Il rangera les instructions en mémoire selon leur ordre croissant et les exécutera dans le même ordre, quel que soit le « pas ». Ainsi, au lieu d'écrire 15 pour l'instruction manquante, on aurait pu mettre 11, ou 12, ou 13, ou 14, ou... 19 : elle se serait placée entre 10 et 20 de la même façon.

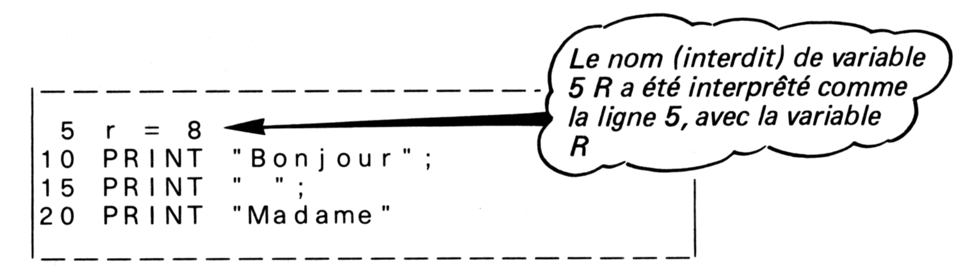
Prenez donc, vous aussi, l'habitude de numéroter les instructions par « pas » de 10.

MODE PROGRAMME

Accessoirement, remarquez autre chose. On vient de voir que le fait de commencer une ligne par un numéro provoque le passage en *mode programmé ;* or, rappelez-vous ce qu'on a dit dans le chapitre précédent, à propos des noms de variables : ils ne doivent *pas* commencer par un chiffre. Dans ce cas, en effet, le Basic imaginerait que vous avez voulu numéroter une ligne. Si vous écriviez :



puis faisiez RETURN, le Basic penserait qu'il s'agit de la ligne 5 avec l'affectation R = 8. Il introduirait donc cette ligne, comme le prouve le listage :




```
 5 r = 8
10 PRINT "Bonjour";
15 PRINT " ";
20 PRINT "Madame"
```

Laissons cette ligne 5, pour l'instant, car elle n'interviendra pas dans l'exécution du programme (qui ne demande pas l'affichage de R).

2. Les corrections

REMPLACEMENT D'UNE LIGNE

Supposons qu'au lieu d'écrire *Bonjour Madame,* on veuille mettre *Bonjour Monsieur :* il faut *modifier* la ligne 2Ø, en créer une autre plus exactement. Frappez-là ainsi :

```
]20 ? "Monsieur"
```

Puis, faites ↵ et LIST, afin de vérifier ce qui s'est passé. Vous lirez :

```
]LIST

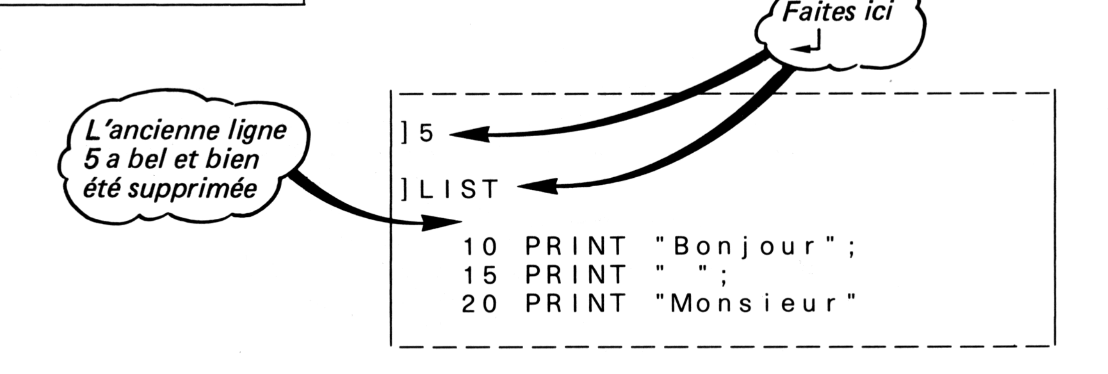
 5 r = 8
10 PRINT "Bonjour";
15 PRINT " ";
20 PRINT "Monsieur"
```

La nouvelle ligne 2Ø *a pris la place* de l'ancienne, qui est perdue.

Pour remplacer une ligne d'instruction par une autre, il suffit de frapper la nouvelle instruction avec le même numéro de ligne.

SUPPRESSION D'UNE LIGNE

En poussant ce principe à son extrême, on peut se demander ce qui se passerait si l'on frappait un numéro de ligne existant, puis *rien* après, sauf le RETURN ? Et bien, la nouvelle ligne *vide* prendrait la place de l'ancienne, l'effacerait. Vérifions-le avec la ligne 5, inutile :




```
]5

]LIST

   10 PRINT "Bonjour";
   15 PRINT " ";
   20 PRINT "Monsieur"
```

Pour supprimer une ligne de programme, il suffit de frapper son numéro puis de faire RETURN (↵).

Rappelez-vous aussi que, pour supprimer une ligne en cours de frappe, donc avant d'avoir fait ↵, on peut frapper CONTROLE/X qui l'annule.

CONTROL/X place une barre oblique inversée à la fin de la ligne en cours, l'annule, et ramène le curseur au début de la ligne suivante.

MESSAGES D'ERREUR

Enfin, notez encore un point important, afin de ne pas être surpris par la suite. Si vous frappez une instruction comportant une erreur, le Basic pourra :

• Soit la détecter aussitôt, dès la validation par ↵ et vous le signaler en refusant l'ordre (par exemple, en affichant un message d'erreur).

• Soit ne pas la détecter au ↵, donc l'accepter, mais s'apercevoir de l'erreur au RUN. Dans ce cas, il vous affichera encore l'un de ces nombreux messages d'erreur dont il a le secret.

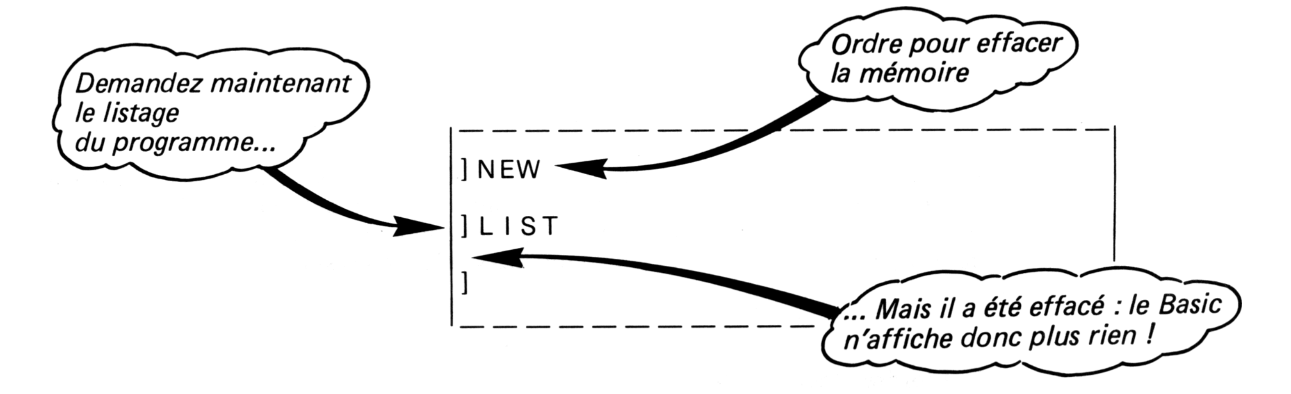
Si, donc, après un RUN l'ordinateur vous indique une erreur (en précisant son numéro de ligne), vérifiez attentivement la ligne indiquée pour la découvrir. Même si cette erreur ne vous saute pas immédiatement aux yeux, restez persuadé que c'est le Basic qui a raison et qu'il y en a bien une.

3. Effacements de la mémoire et de l'écran

EFFACEMENT DE LA MEMOIRE

NEW

On a déjà introduit plusieurs ordres en mémoire ; comment les effacer en bloc, toutes les lignes d'un coup ? Une méthode radicale consiste à faire un RESET COMPUTER.
Une méthode plus élégante consiste à signifier à l'ordinateur qu'on veut libérer les mémoires pour créer un nouveau programme. C'est l'ordre NEW, pour *nouveau ;* « entrez-le » en mode direct (le mot *entrer* fait partie du jargon de l'informatique ; il signifie : *frapper au clavier puis valider avec* ↵). Si, aussitôt, vous demandez un LIST, vous n'obtiendrez plus rien :



Les valeurs que vous aviez tapées ou que le programme avait calculées ont disparu aussi, mais l'écran reste bien rempli ; profitons-en pour l'effacer. Vous souvenez-vous comment ? Oui, bravo : avec CONTROLE/L. Maintenez CONTROLE enfoncée et frappez L : tout s'efface et le curseur revient à l'angle supérieur gauche.

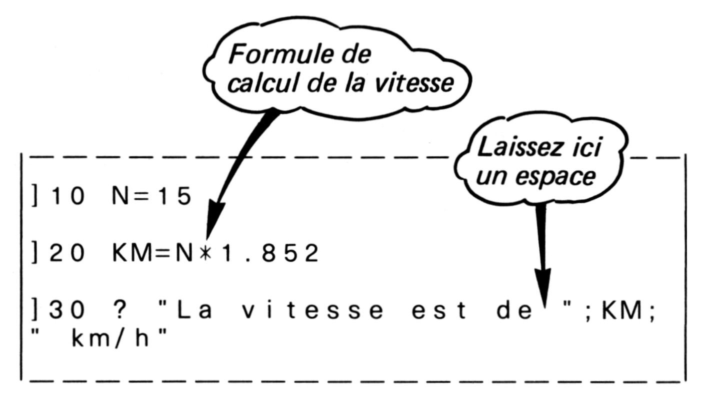
4. Quelle est la vitesse du bateau ?

PROGRAMME DE CONVERSION

On va profiter de ce bel écran pour rédiger un programme de caractère utilitaire : à quoi correspond, en kilomètres par heure, la vitesse d'un bateau donnée en *nœuds ?* Dire qu'un bateau « file 15 nœuds » signifie qu'il se déplace à la vitesse de 15 *milles marins* à l'heure, un *mille marin* valant 1 852 mètres (toutes choses que vous saviez). Le programme suivant va convertir les nœuds en kilomètres-heure ; il consiste à multiplier par 1,852 la vitesse en nœuds.

> **Rappelez-vous qu'en Basic, la virgule décimale est remplacée par un point décimal et que le symbole de la multiplication est l'astérisque (*).**

Entrez en machine le programme suivant, où N est le nom de variable de la vitesse, en nœuds, et KM le nom de variable de la vitesse en km/h :




```
]10 N=15
]20 KM=N*1.852
]30 ? "La vitesse est de ";KM;
" km/h"
```

En frappant ce programme, rappelez-vous que le texte des messages entre guillemets peut être en minuscules ou en majuscules : ces caractères seront respectés par le Basic.

A la ligne 10, on attribue la constante 15 à la variable N (nœuds) ; en 20, on calcule la vitesse en kilomètres-heure, mais on n'affiche toujours rien. Ce n'est qu'en 30 qu'on commande l'affichage PRINT (avec ?) d'une formule complexe : tout ce qui est placé « entre guillemets » va être reproduit tel, mais au centre, KM sera remplacé par sa valeur.

Lancez ce programme avec RUN et vous obtiendrez en-dessous :



```
]RUN
La vitesse est de 27.78 km/h
```

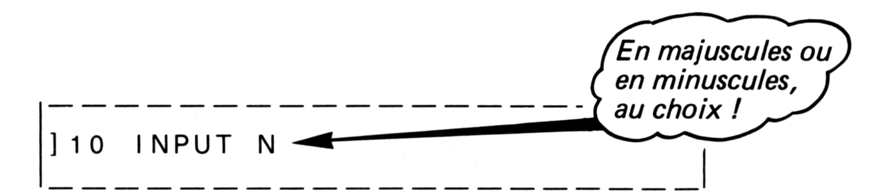
En effet, 15 nœuds correspondent bien à 27,78 km/h, puisque c'est l'ordinateur qui le dit. Essayez d'autres vitesses en nœuds, en refrappant la ligne 10 et en mettant d'autres valeurs à la place de 15. Puis, quand vous en aurez assez de refrapper cette ligne, nous vous montrerons une façon plus intéressante de procéder.

5. L'ordre d'entrée INPUT

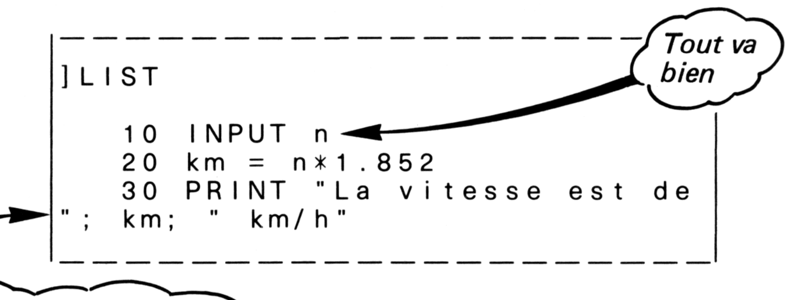
INPUT

Cette fois, on va confier au programme le soin de vous interroger sur la vitesse en nœuds choisie. Pour cela, on remplace la ligne 10 par une nouvelle ligne comportant un ordre de « demande d'entrée d'une nouvelle donnée », ce qui se dit INPUT *(entrez)* en Basic.

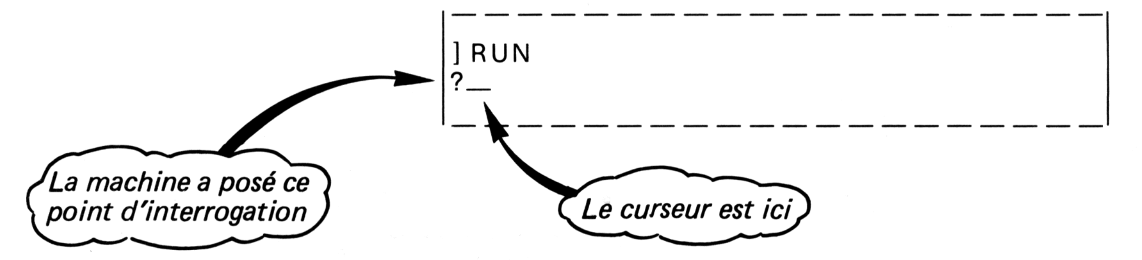
Vous vous souvenez que, pour remplacer une ligne, il suffit de refrapper la nouvelle avec le même numéro. Faites donc, en frappant les cinq lettres de INPUT :



Vérifiez, avec LIST, que la substitution s'est bien faite :



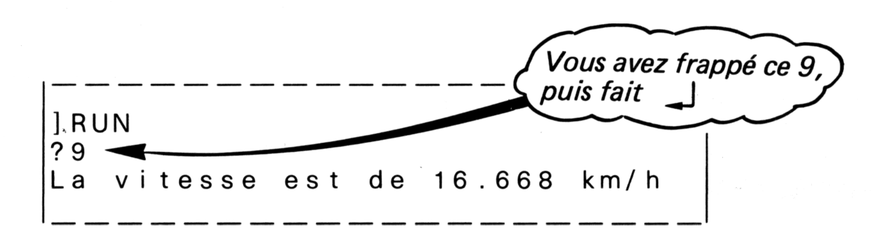
Effacez donc à nouveau l'écran avec CONTROLE/L, puis lancez un RUN et observez ce qui s'affiche : un simple point d'interrogation au début de la ligne ! Cela donne :



En effet, lorsque le Basic rencontre un ordre INPUT (ce qu'il vient de faire), il affiche un ? et stoppe toute activité, attendant que vous frappiez une donnée au clavier que vous validerez avec un ↵. Dès lors, il poursuit l'exécution du programme.

Attention : ne confondez pas le ? affiché par le Basic qui vient de rencontrer un INPUT avec le ? que vous placez pour remplacer PRINT. C'est bien le même symbole, mais pour deux choses différentes.

En réponse à la demande du programme, frappez par exemple 9, pour voir, puis faites ↵ ; l'affichage complet sur l'écran sera :



Vous pouvez frapper à nouveau sur RUN et introduire d'autres valeurs, l'essentiel étant que vous compreniez parfaitement le fonctionnement de INPUT.

6. Et si l'on recommençait ?

Si vous aviez de multiples conversions à exécuter, les frappes successives de RUN pour relancer le programme seraient fastidieuses. Dans ce cas, il est plus simple d'ajouter une nouvelle instruction, à la fin de ce programme, qui lui ordonne de recommencer en renvoyant à son début.

GOTO

Cette nouvelle instruction portera le numéro 40. Elle ordonnera au programme d'aller sans discussion à l'instruction 10, donc de revenir à son début. *Aller à* se dit *go to,* en anglais, et s'écrit en un seul mot GOTO. Entrez :

```
]40 GOTO 10
```

«Aller obligatoirement à 10»

Faites LIST, pour voir où vous en êtes :

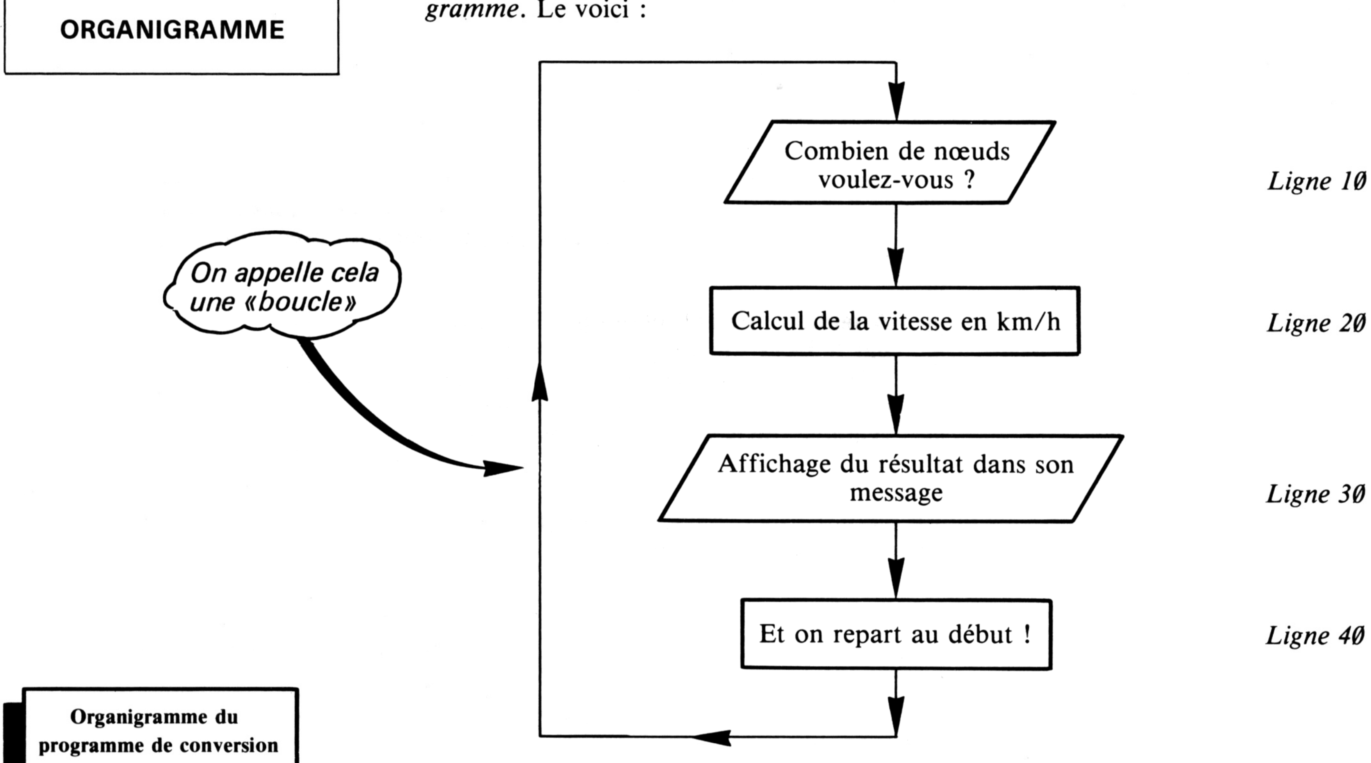
```
]LIST

   10 INPUT n
   20 km = n*1.852
   30 PRINT "La vitesse est de
"; km; " km/h"
   40 GOTO 10
```

Remarquez que le Basic a, cette fois, remis en minuscules tous les noms des variables

ORGANIGRAMME

On peut représenter ce programme par un petit dessin qu'on appelle un *organigramme*. Le voici :



Organigramme du programme de conversion

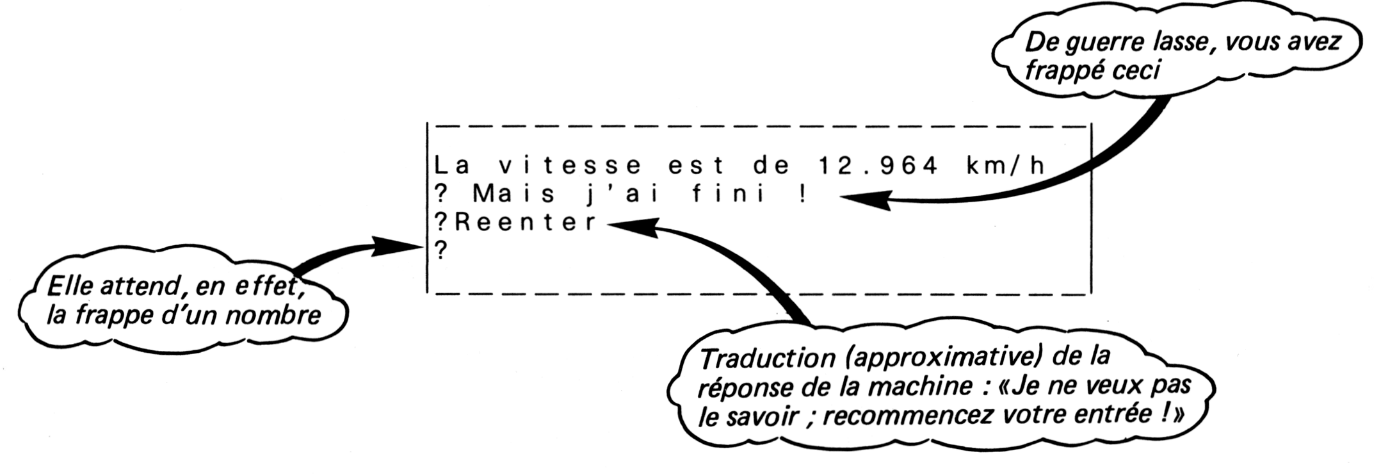
Au passage, notez que l'organigramme répond à quelques règles de dessin qui veulent que les ordres d'*entrée* (INPUT) ou de « sortie » vers l'écran (PRINT) soient inscrits dans des parallélogrammes, alors que les ordres d'exécution « internes » à la machine figurent dans des rectangles.
Toujours est-il que cet organigramme montre bien que, de la fin du programme, on repart à son début en une « boucle ». Faites RUN ; la machine va vous interroger une première fois : supposons que vous frappiez 9. Elle va afficher le résultat puis vous ré-interrogera à nouveau : on suppose que vous frappez 11 ; puis ça continue, et vous frappez 7... Cela donnera sur l'écran :

```
]RUN
?9
La vitesse est de 16.668 km/h
?11
La vitesse est de 20.372 km/h
?7
La vitesse est de 12.964 km/h
?
```

La machine ne se lasse pas et en redemande... On suppose que vous avez fini vos conversions

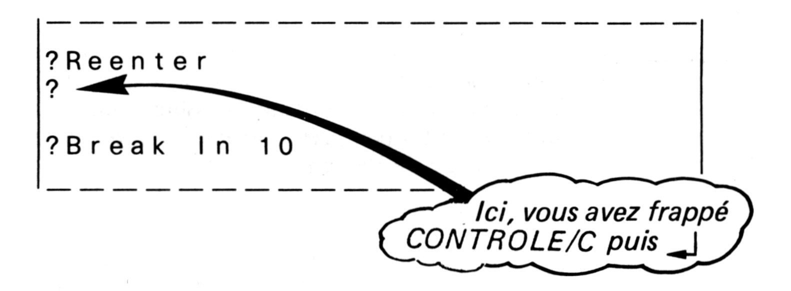
REENTER

En effet, elle obéit aveuglément au programme et ne saura pas que vos calculs sont terminés. Si vous frappiez n'importe quoi d'autre que des chiffres, elle serait interloquée :



CONTROLE/C

Le message « Reenter » signifie, en effet : « Refaites votre entrée ». Comment indiquer à cette obstinée que nous en avons terminé ? Simplement en frappant la combinaison CONTROLE/C puis en faisant ↵, ce qui donnera, si l'on reprend la fin de l'écran précédent :



La machine est, désormais, prête à accepter d'autres ordres, même un nouveau RUN pour relancer le même programme.

En général, lorsqu'on utilise une instruction INPUT, on s'arrange pour qu'un message explicatif précède le point d'interrogation sur l'écran. Vous obtiendrez ce résultat en créant une nouvelle ligne, 5 par exemple ; mais il faudra *aussi* que le GOTO de la ligne 40 renvoie en 5 et non plus en 10, ce qui va vous obliger à refrapper ces deux lignes :

```
]5 ? "Donnez la vitesse en noeu
ds:"

]40 GOTO 5
```

De ce fait, le programme complet, suivi de son lancement, donne :

```
]List

  5 PRINT "Donnez la vitesse
en noeuds:"
  10 INPUT n
  20 km = n*1.852
  30 PRINT "La vitesse est de
"; km; " km/h"
  40 GOTO 5

]RUN
Donnez la vitesse en noeuds:
?6.75
La vitesse est de 12.501 km/h
Donnez la vitesse en noeuds:
?

?Break In 10
```

Etc, jusqu'à un CONTROLE/C

Faites un CONTROLE/C puis un ↵ pour stopper ce programme.

BRANCHEMENT INCONDITIONNEL

Notez encore que l'ordre GOTO commande un branchement « inconditionnel », sans discussion, à la ligne indiquée. C'est pourquoi on l'appelle « branchement inconditionnel ».

7. Effacement programmé de l'écran

EFFACEMENT PROGRAMME DE L'ECRAN

HOME

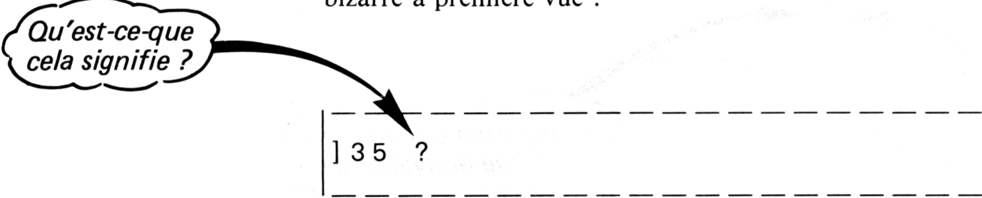
On peut fignoler ce programme en commençant, dès le RUN, par effacer l'écran. Oui, mais comment introduire un CONTROLE/L dans le programme ? Différentes méthodes sont possibles ; en voici une : vous commandez HOME en anglais, soit à peu près : *retour à la maison,* c'est-à-dire renvoi du curseur en haut et à gauche de l'écran avec effacement de celui-ci.

Attention : la touche HOME marquée aussi R ne fait que renvoyer le curseur à l'angle supérieur gauche. L'ordre HOME, lui, efface l'écran.

Vérifiez-le tout d'abord en frappant les quatre lettres HOME sur le clavier. Faites ensuite un ↲ et vous constaterez que l'écran s'efface totalement. Puisque ça fonctionne, introduisons cette instruction dans le programme, par exemple en ligne 3 :

```
]3 HOME
```

Dès lors, et avec le RUN, *le programme commence par effacer l'écran ;* vous n'y trouverez plus que les lignes dont vous avez commandé l'affichage.
Un dernier défaut subsiste encore : les réponses ne sont pas séparées de nouvelles questions ; introduisons un saut de ligne, d'un calcul à l'autre, avec cet ordre bizarre à première vue :



POUR SAUTER UNE LIGNE

Ici, on a frappé un numéro de ligne suivi par un ordre PRINT (? en abrégé), suivi lui-même par... rien du tout ! Dans ce cas, le Basic va lire ce PRINT et s'apercevra qu'il ne doit rien afficher ; par conséquent, il sautera une ligne. Revoyons le programme complet, d'abord :

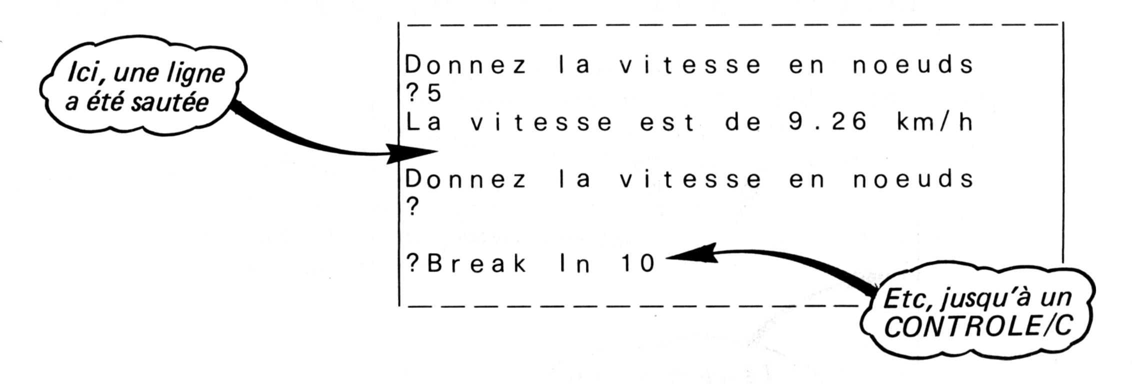
```
]list

   3 HOME
   5 PRINT "Donnez la vitesse
en noeuds"
  10 INPUT n
  20 km = n*1.852
  30 PRINT "La vitesse est de
"; km; " km/h"
  35 PRINT
  40 GOTO 5

]RUN
```

Pour sauter une ligne avant de recommencer

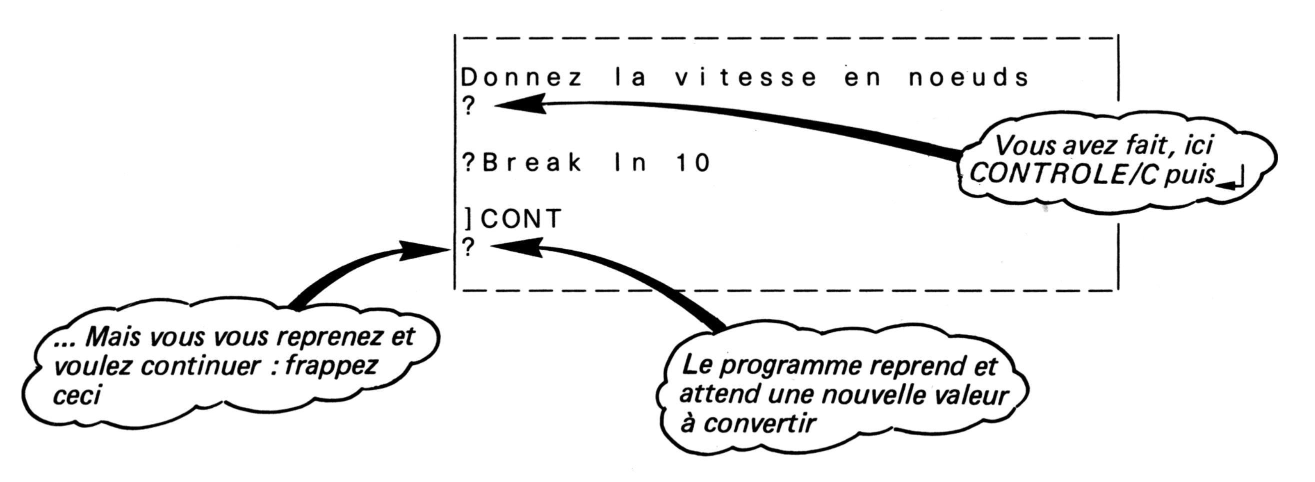
Avec le RUN puis le ↲, vous verrez :



Notez bien le rôle de ce PRINT suivi par rien du tout (si ce n'est ↲) : il sert à sauter une ligne.

CONT

Deux PRINT successifs auraient servi à sauter deux lignes, par exemple.
Pour en terminer avec ce programme, vous faites un CONTROLE/C mais, pris de remords, vous voulez continuer vos conversions : que faire ? Frappez simplement CONT, pour continuer, et il reprendra au point exact où vous l'aviez lâché. La séquence serait la suivante :



Dès lors, vous pouvez effectivement continuer.

8. Le code ASCII

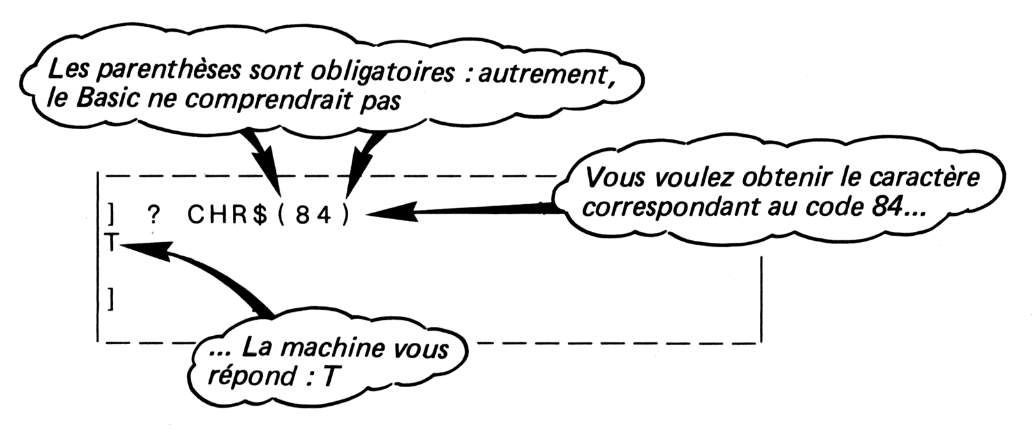
Sachez une chose importante : chaque caractère que vous pouvez frapper au clavier, qu'il soit simple ou combiné avec SHIFT ou CONTROLE, dispose d'un code numérique allant de 0 à 255.

CODE ASCII

L'ensemble de ces codes est normalisé (plus ou moins...) et relève des normes ASCII, ce qui se prononce *Aski* et vient de « *American Standard Code for Information Interchange* ». Les normes strictes ASCII, allant de 32 à 127, sont données en fin de ce livre ; aux autres codes correspondent des ordres divers ou des caractères graphiques sur lesquels on reviendra.

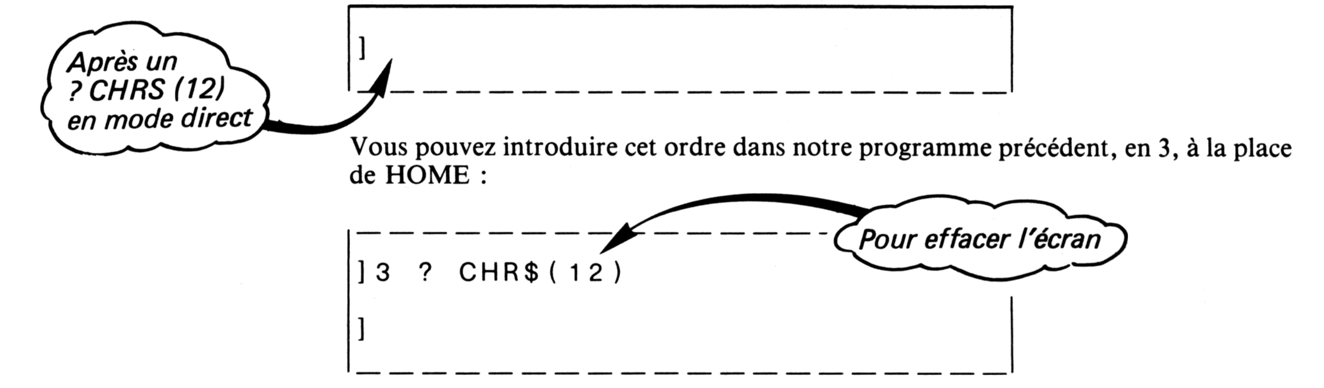
CHR$

On peut demander au Basic un caractère quelconque en frappant son code. Sachant que *caractère* se dit « *character* » en anglais, on comprendra que l'ordre se code CHR ; en outre, il doit être suivi du fameux symbole *dollars* ($) puisqu'il s'agit de caractères précisément. On écrira donc PRINT CHR$ suivi du code du caractère voulu. Le T, par exemple, a pour code 84, ce que vous pouvez vérifier dans le tableau ASCII ; pour afficher un T, vous pourrez écrire (faites-le en mode direct) :



Tentez votre chance avec d'autres valeurs ; par exemple, le *t* minuscule se code 116 : il y a toujours une différence de 32 entre majuscules et minuscules.

Or, l'effacement qui se frappe avec CONTROLE/L, possède aussi son code : c'est le 12. Vous devinez alors que l'on puisse aussi commander l'effacement avec ? CHR$(12). Vérifiez-le en mode direct : l'effet sera semblable à celui de HOME, le caractère d'appel du Basic venant alors toutefois sur la 3e ligne et non sur la seconde comme avec HOME :

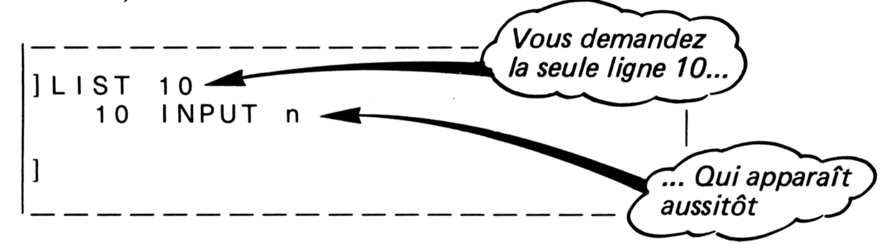


Vous pouvez introduire cet ordre dans notre programme précédent, en 3, à la place de HOME :

Relancer le programme : vous n'y verrez guère de différence, et vous n'en verrez plus du tout si vous ajoutez un point-virgule au bout de la ligne 3, puisque grâce à lui vous supprimez un retour à la ligne !

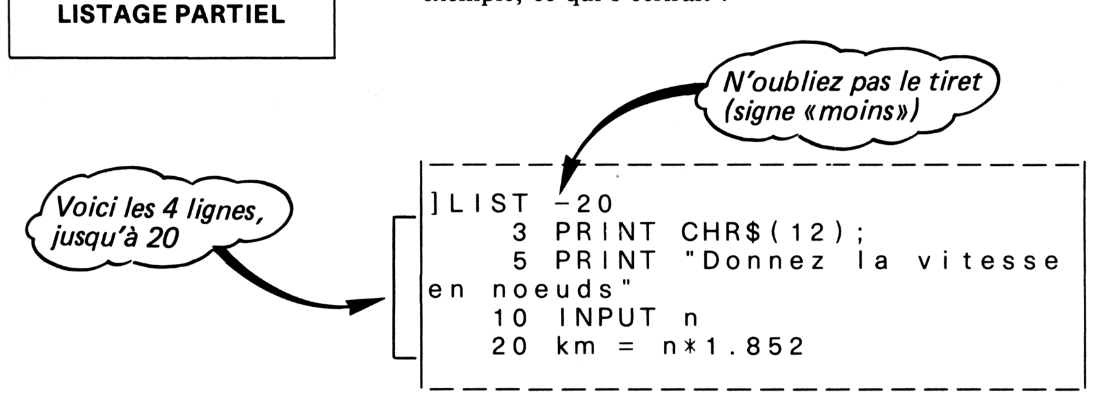
9. Travaillez sur les listages

Vous l'apprendrez vite à vos dépens, la mise au point d'un programme fonctionnant comme on l'espérait impose bien des retouches et des corrections. Nous avons déjà passé en revue certaines méthodes, mais il nous faut d'emblée aller encore un peu plus loin.
Vous serez amené à vous servir fréquemment de ces diverses méthodes ; c'est pourquoi nous vous invitons à lire ce passage, à expérimenter ce que nous vous proposerons, puis à y revenir dès lors que le besoin s'en fera sentir.
Considérons l'ordre LIST. Vous pouvez très bien *lister* une seule ligne, par exemple la ligne 10 du programme précédent (nous supposons qu'il est toujours en mémoire) :

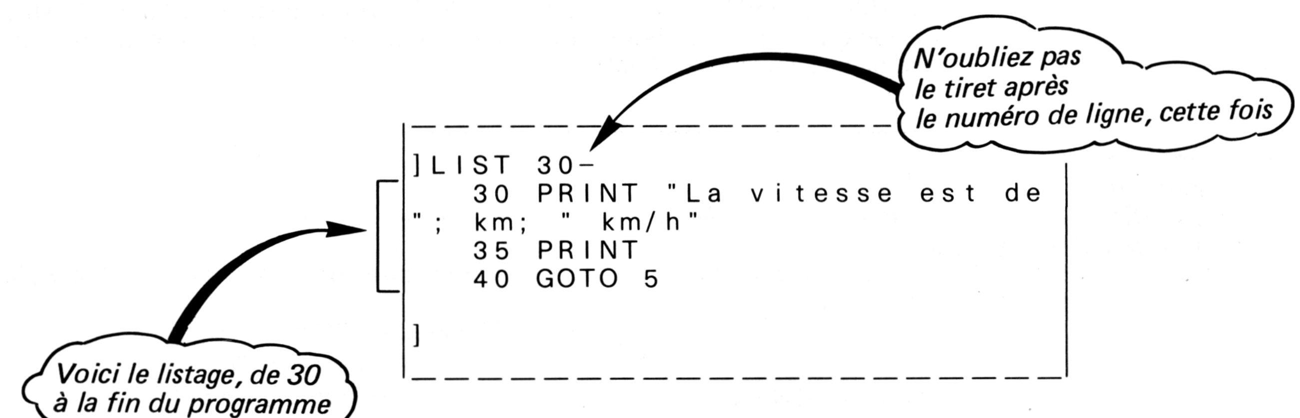


LISTAGE PARTIEL

Vous pourriez demander le listage du programme jusqu'à la ligne 20 incluse, par exemple, ce qui s'écrirait :



Ou encore, vous pourriez souhaiter les lignes à partir de 30 incluse :




```
]LIST 30-
   30 PRINT "La vitesse est de
"; km; " km/h"
   35 PRINT
   40 GOTO 5

]
```

Dernière formule, enfin, le listage à partir de 5, par exemple, jusqu'à 20 inclus ; voici :

```
]LIST 5-20
    5 PRINT "Donnez la vitesse
en noeuds"
   10 INPUT n
   20 km = n*1.852
```

MODIFICATION D'UNE LIGNE

Supposons maintenant que vous vouliez modifier la ligne 40 et écrire à la place GOTO 10 (au fait, pouvez-vous deviner quel sera le résultat de cette modification ?). Vous pourriez refrapper totalement cette ligne (pourquoi pas ?). Mais il existe une méthode souvent préférable permettant d'apporter des modifications à une ligne sans avoir à la refrapper. Pour cela, il faut « éditer » cette ligne, ce qui signifie dans ce jargon très discutable de l'informatique : l'afficher (ou l'imprimer). C'est là où LIST intervient. Notre programme étant court, vous pourriez aussi bien le lister en entier. Puis :

1) — Servez-vous des touches fléchées (autour de la barre d'espacement) pour emmener le curseur tout au début de la ligne 40. Voici ce que cela donnera (on ne vous montre que cette seule ligne) :

```
]LIST 40
  -40 GOTO 5
```

Vous avez amené le curseur, ici,

RECOPIE D'UNE LIGNE

2) — Servez-vous, maintenant, de → pour *déplacer le curseur vers la droite.* Avec →, le curseur « relit » les caractères qu'il croise et, pour l'ordinateur, tout se passe comme si vous les refrappiez. Stoppez le curseur sur le 5.

3) — Maintenant, frappez votre correction, donc un 10 au lieu du 5.

Attention : seul → relit et « recopie » en mémoire les caractères sur lesquels il passe. Cela, en mode « édition ».

4) — Faites ↲ : la nouvelle ligne est enregistrée.

5) — La correction est terminée, mais il reste encore une précaution à prendre : extrayez le curseur du corps du programme si vous avez listé celui-ci en entier, emmenez-le au-dessous avec la touche marquée d'une flèche verticale (↓) car sinon, ce que vous frapperiez à nouveau viendrait se conjuguer avec les lignes existantes, donnant un résultat imprévisible. Ou alors, et plus simplement, effacez l'écran avec CONTROLE/L.
Vous pouvez maintenant vérifier l'état de la nouvelle ligne 40 modifiée, avec LIST, ou simplement LIST 40 ; vous obtiendrez :

```
]LIST 40
   40 GOTO 10

]
```

Que pensez-vous que cela donnera, avec un RUN ? On ne repose plus la question « Donnez la vitesse en nœuds » à chaque tour ; ce qui ne nous avance guère avec ce programme !

Questions sur le chapitre IV

1. Y a-t-il une erreur dans la séquence suivante ?

```
1 ? "Bonjour,"
2 ? "chers amis."
999 ? "A bientôt."
```

2. Pour lancer l'exécution d'un programme, l'ordre est :
   - ☐ RUN
   - ☐ GO

3. Un point-virgule placé dans un ordre PRINT :
   - ☐ Commande un retour à la ligne
   - ☐ Supprime le retour à la ligne

4. Peut-on ré-afficher un programme se trouvant en mémoire ?
   - ☐ Oui, avec l'ordre LIST
   - ☐ Non, il faut le refrapper

5. Pour supprimer une ligne de programme, il faut :
   - ☐ Frapper son numéro, puis faire RETURN ↵
   - ☐ Frapper son numéro, puis faire CONTROLE/X

6. L'ordre NEW sert à :
   - ☐ Relancer l'exécution d'un programme
   - ☐ Effacer la mémoire

7. Pour effacer l'écran, il faut faire :
   - ☐ HOME
   - ☐ CONTROLE/L
   - ☐ ? CHR$(12)

8. Si le programme doit demander à l'opérateur de nouvelles données, vous devrez introduire une instruction :
   - ☐ ENTER
   - ☐ INPUT

9. Un organigramme est :
   - ☐ La représentation graphique d'un algorithme
   - ☐ Le dessin d'un programme

10. Pour interrompre un programme en cours d'exécution, vous ferez :
   - ☐ CONTROLE/C
   - ☐ STOP

Réponses page 90

CHAPITRE V

DES BOUCLES ET DES COULEURS

Une notion importante en programmation est celle de boucle. Après en avoir traité avec l'ordre GOTO, on va étudier ici la création de boucles avec FOR... TO... NEXT et les règles qui s'y attachent. On appliquera ces notions à l'usage de tracés simples en couleurs, parce qu'enfin, on va commencer à travailler avec la palette de votre Adam.

Principaux thèmes étudiés

Boucles	Mode graphique basse résolution	END
FOR... TO	GR	Rectangle
NEXT	Grille graphique GR	Remarques
Organigramme de boucle FOR... TO	TEXT	REM
Instructions multiples	Couleurs	Diagonale
Double-point	COLOR	Fond couleur
Pas dans FOR... TO	PLOT	Emboîtage des boucles
STEP	Trait horizontal	Règles avec les boucles
Décomptage avec FOR... TO	Run n° de ligne	HLIN
	Trait vertical	VLIN
		Pointillé

1. Boucles avec FOR... TO

BOUCLES

On a vu, dans le chapitre précédent, que l'ordre GOTO pouvait commander un « bouclage ». L'ordre FOR... TO permet également la création de boucles, mais avec deux différences essentielles :

1) — Voici le principal : vous contrôlez le *nombre de boucles* à effectuer, puisque FOR... TO en fixera les limites.

FOR... TO

NEXT

2) — Puis, mais c'est le moins important pour notre action, cette instruction se compose en réalité de deux ordres distincts. L'un est FOR... TO ce qui signifie « pour... jusqu'à » et sert à fixer des limites, le second est NEXT (« au suivant ! »), qui renvoie à la valeur suivante. Il faudra donc *deux* lignes.

C'est facile à vérifier avec ce court programme qui va afficher les trois chiffres 1, 2 et 3. Entrez-le en machine et exécutez-le pour voir ce que cela donne ; on vous l'expliquera aussitôt après ; on a fait suivre l'exécution, ici :

```
]10 FOR N=1 TO 3

]20 ? N

]30 NEXT N

]40 ? "C'est fini !"

]RUN
1
2
3
C'est fini !

]
```

Que s'est-il passé ? A la ligne 1Ø, on a fixé les limites de N, qui pourra aller de 1 à 3, en commençant par 1.

Le comptage se fait par unités successives : 1, puis 2, puis 3... A moins qu'on n'ajoute des ordres contraires, comme on va le voir plus loin.

Puis, en 2Ø, on commande l'affichage de N, donc de 1 au premier tour. Le programme rencontre alors la ligne 3Ø qui lui demande le N suivant ; il se branche alors à nouveau en 1Ø et, s'apercevant que la valeur 1 a déjà été utilisée, prend la 2 et l'attribue à N. En 2Ø, il affiche 2, Puis passe en 3Ø qui renvoie en 1Ø où N prend la valeur 3. Celle-ci est affichée à son tour, puis 3Ø renvoie encore en 1Ø mais là, le programme s'aperçoit que toutes les valeurs possibles ont été utilisées. Par conséquent, il saute toute la chaîne et poursuit à l'instruction suivante qui est la 4Ø ; il affiche alors « C'est fini ! ».

Lorsque toutes les valeurs permises ont été utilisées, le programme passe à l'instruction qui suit la ligne NEXT.

ORGANIGRAMME DE LA BOUCLE FOR... TO

On peut illustrer ce processus à l'aide de l'organigramme suivant ci-contre.
Remarquez le losange de l'organigramme : il pose une question, contenue implicitement dans l'ordre FOR... TO : *la valeur finale est-elle dépassée ?* si *non,* on boucle ; si *oui,* on passe à l'instruction suivante. On emploie donc deux sommets de ce losange, selon que la réponse est NON ou OUI. Chaque fois qu'on posera une question, on l'insérera dans un losange.
L'instruction de traitement contenue dans la boucle peut fort bien faire appel à autre chose que la variable contenue dans la boucle FOR... TO ; modifiez simplement le programme précédent à la ligne 20 pour obtenir :

```
    10 FOR n = 1 TO 3
    20 PRINT "Oui,";
    30 NEXT N
    40 PRINT "C'est fini !"

]RUN
Oui,Oui,Oui,C'est fini !
```

Boucle FOR ... TO

Initialisation de la boucle : on fixe les valeurs limites *de... à* soit FOR... TO de la variable (N, ici) — *Ligne 10*

Instruction de traitement : ici, l'affichage de N — *Ligne 20*

Valeur suivante de N — *Ligne 30*

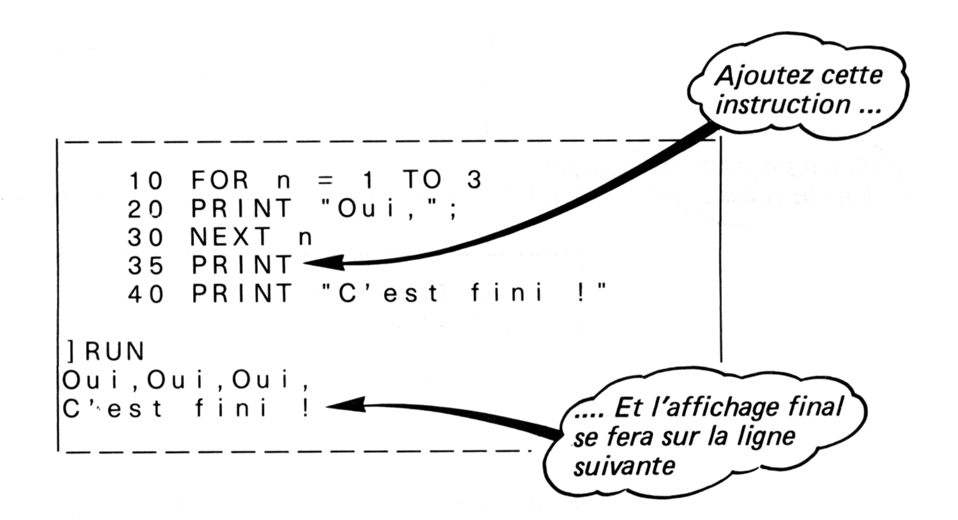
Question implicite de FOR ... TO

Dépasse-t-on la valeur finale ? — NON / OUI

Affichage final de C'est fini ! — *Ligne 40*

Organigramme de la boucle FOR... TO

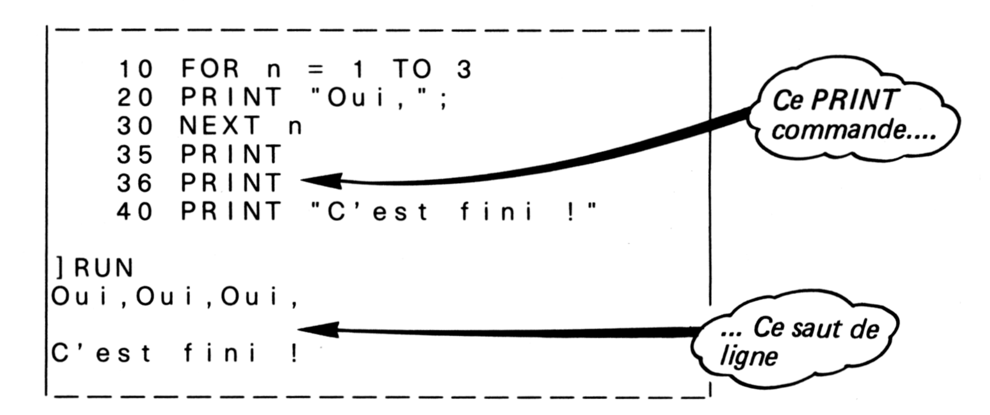
Comment faire pour écrire C'est fini ! sur la ligne suivante ? Réfléchissez un instant : il faut *annuler* l'effet du point virgule prédécent, de la ligne 20, mais *seulement* lorsqu'on sort de la boucle. Qu'à cela ne tienne : on va placer un PRINT rien du tout, *sans* point-virgule final, lui ; on obtiendra cet affichage sur *la même ligne* que les OUI,OUI,OUI, mais il provoquera un retour à la ligne pour l'affichage final. Cela donne, en récapitulant :




```
   10 FOR n = 1 TO 3
   20 PRINT "Oui,";
   30 NEXT n
   35 PRINT
   40 PRINT "C'est fini !"

]RUN
Oui,Oui,Oui,
C'est fini !
```

Et comment pourriez-vous faire pour séparer d'une ligne vide les « OUI,OUI,OUI », de « C'est fini ! » ? On nous l'a déjà dit dans le chapitre précédent, rappelez-vous : placez un autre PRINT :




```
    10 FOR n = 1 TO 3
    20 PRINT "Oui,";
    30 NEXT n
    35 PRINT
    36 PRINT
    40 PRINT "C'est fini !"

]RUN
Oui,Oui,Oui,

C'est fini !
```

Mais pourquoi employer *deux* lignes, 35 et 36, qui ne comprennent pas grand chose chacune ? Ne pourrait-on les regrouper ? Oui, le Basic accepte qu'on place *plusieurs* instructions sur une unique ligne (avec un *unique numéro*).

INSTRUCTIONS MULTIPLES

DOUBLE-POINT

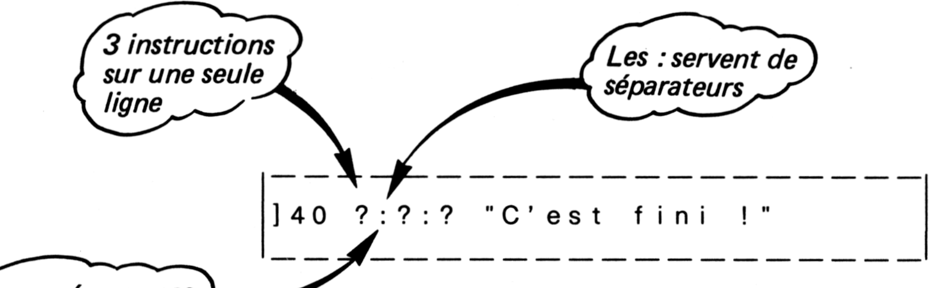
Pour séparer des instructions sur une ligne, on place un double-point (:). C'est le « séparateur » officiel.

Aussi, on peut ré-écrire encore ainsi le programme précédent, en cumulant même 3 lignes. Commencez par *supprimer* 35 et 36, puis refrappez ainsi la ligne 40. Pour cela, frappez 35, puis ↵, et 36 suivi d'un ↵.

```
]35

]36
```

Puis frappez :




```
]40 ?:?:? "C'est fini !"
```

Un LIST vous donnera le programme complet suivant :

```
    10 FOR n = 1 TO 3
    20 PRINT "Oui,";
    30 NEXT n
    40 PRINT: PRINT: PRINT "C'es
t fini !"
```

Le RUN, bien entendu, procurerait le même résultat que ci-dessus.

PAS DANS FOR... TO
STEP

2. Boucles FOR... TO avec un pas

On a vu que la boucle FOR... TO « incrémentait » la variable d'un *pas positif unitaire :* la variable N passait successivement à 1, puis 2, puis 3. Or, on peut lui imposer un *pas* différent, par exemple un demi, ce qui s'écrit Ø,5 en français courant.

> **Or, rappelez-vous que le Basic, d'origine anglaise, place un « point » décimal là où nous mettons une « virgule » décimale. Donc, au lieu d'écrire 0,5 on écrira 0.5.**
> **Bien mieux, le Basic estime que le 0 avant le point est encore superflu. On peut donc se contenter d'écrire .5.**

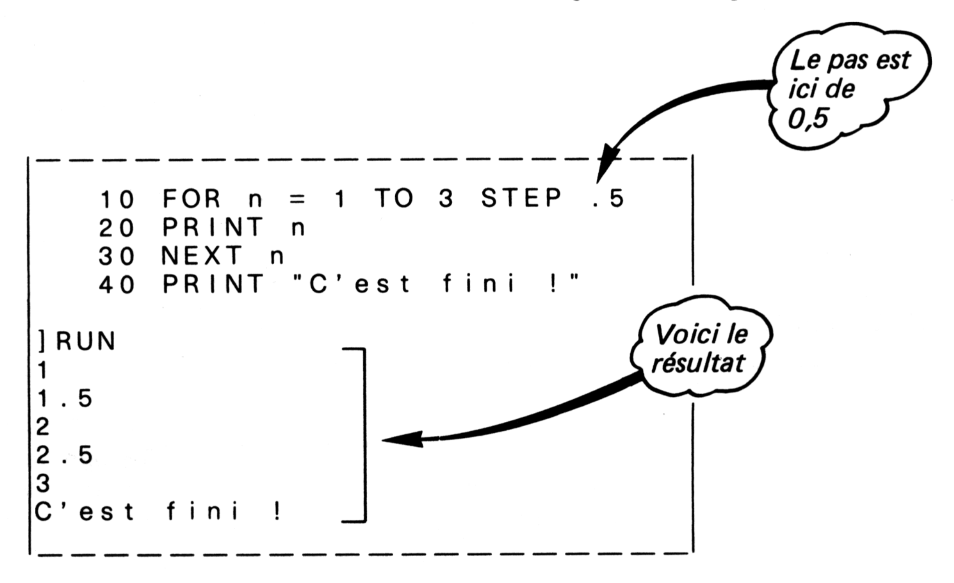
Si vous placez cependant un 0, cela ne constitue pas une faute.
Avez-vous conservé en mémoire le programme précédent ? Si oui, commencez par modifier la ligne 10 pour obtenir un pas de .5, ce qui se dit STEP.5 (« step » signifie *pas,* en anglais). Pour apporter cette correction, on va employer LIST. Faites successivement.

- LIST 10, puis RETURN. La ligne 1Ø apparaît sur l'écran.
- Avec ↑, amener le curseur après le 3 de "FOR N=1 TO 3".
- Frappez alors STEP .5, puis RETURN.

Tout ceci donne.

```
10 FOR n = 1 TO 3 STEP .5
```

La correction est achevée. Modifiez alors les lignes 2Ø et 4Ø pour obtenir :



```
   10 FOR n = 1 TO 3 STEP .5
   20 PRINT n
   30 NEXT n
   40 PRINT "C'est fini !"
]RUN
1
1.5
2
2.5
3
C'est fini !
```


Vous pourriez choisir un pas absolument quelconque, par exemple de 2. Modifiez à nouveau la ligne 1Ø ainsi, et lancez le programme ; cela donnera :

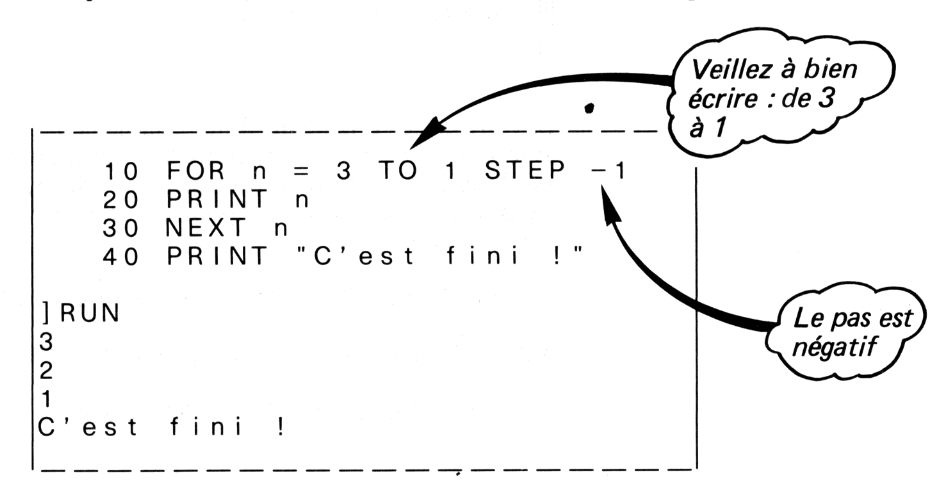
```
    10 FOR n = 1 TO 3 STEP 2
    20 PRINT n
    30 NEXT n
    40 PRINT "C'est fini !"

]RUN
1
3
C'est fini !
```

On a compté par 2

DECOMPTAGE AVEC FOR... TO

Le pas peut être négatif, si l'on veut *décompter,* mais il faudra alors aller de *3 à 1* et non plus de 1 à 3 ; il suffit, à nouveau, de ne modifier que la seule ligne 1Ø :



Vous pouvez combiner tout ceci à volonté, pas quelconques, positifs ou négatifs. Faites quelques expériences par vous même, pour bien « vivre » cet ordre.

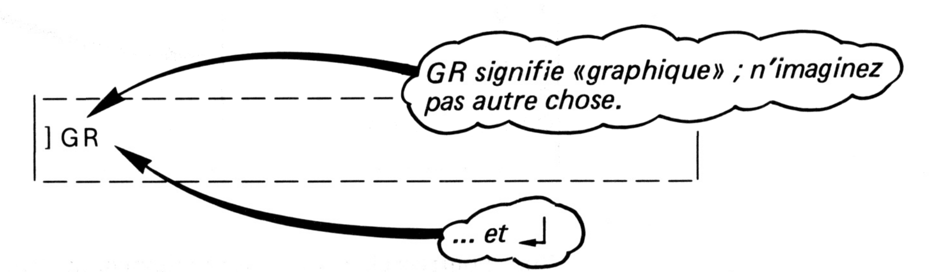
N'oubliez pas, lorsque vous changez le programme, d'effacer l'ancien au préalable avec un NEW.

Faute de quoi, vos instructions se mêleront et vous procureront des résultats inattendus. Rappelez-vous que l'ordinateur a une mémoire d'éléphant, qu'il n'oublie aucune instruction, n'interprète en aucune façon vos désirs, et que, s'il peut vous jouer un tour, il le fera (en toute innocence). On ne vous le rappellera plus !

3. Le graphique « basse résolution »

MODE GRAPHIQUE (BASSE RESOLUTION)

On va, enfin, traiter de la couleur et du graphique, mais en introduction seulement. Votre Basic commande deux modes dits « graphiques », le graphique *basse résolution* et le graphique *haute résolution* où le tracé est beaucoup plus fin. Pour passer au mode graphique basse résolution, il faut commander en mode direct GR puis faire RETURN :



GR

L'écran va passer en mode graphique et s'efface.

• En mode texte, l'écran est divisé en 24 lignes de 31 caractères.
• En mode graphique (basse résolution), les 20 lignes supérieures sont réservées au graphique, les 4 lignes inférieures servant toujours au texte.

GRILLE GRAPHIQUE

Ces 20 lignes de 40 colonnes peuvent être représentées sous forme d'une *grille graphique.* Les colonnes seront numérotées de 0 à 39 (ce qui fait bien 40) et les rangées aussi. Ainsi, on peut viser un « pavé » précis à l'intersection d'une rangée et d'une colonne ; remarquez qu'en raison des dimensions de l'écran, chaque pavé n'est pas carré mais rectangulaire (plus large que haut). Ainsi, on pourra définir les :

- pavé 0,0 à l'extrémité supérieure gauche,
- pavé 39,0 à l'extrémité supérieure droite,
- pavé 0,39 à l'extrémité inférieure gauche,
- pavé 39,39 à l'extrémité inférieure droite,
- un pavé très central pourra être le 20,20.

Entraînez-vous à situer ces pavés sur la grille que nous avons dessinée (voir page suivante). Faites-en des copies pour établir des dessins.

TEXT

Une fois en mode graphique et pour revenir en mode texte (donc, écrire sur tout l'écran), il faudra frapper en mode direct :

```
]TEXT
```


COULEURS

Revenons au mode graphique : Adam vous permet de choisir la couleur de l'affichage. En effet, 16 couleurs sont disponibles et leurs codes sont les suivants :

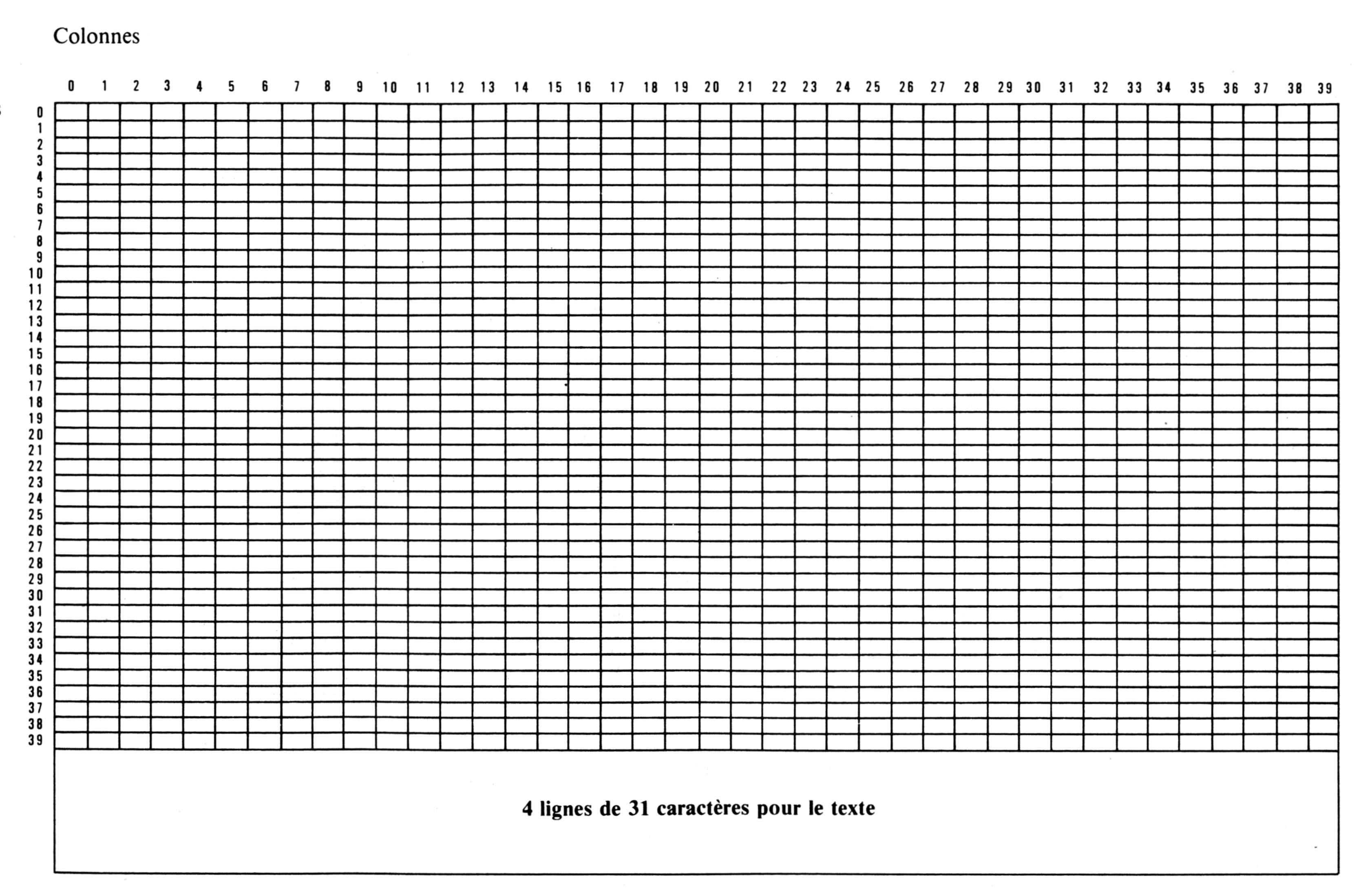
Code	Couleur	Code	Couleur
0	Noir	8	Jaune
1	Magenta	9	Rouge
2	Bleu foncé	10	Gris 2
3	Pourpre	11	Rose
4	Vert foncé	12	Vert clair
5	Gris 1	13	Jaune clair
6	Bleu moyen	14	Vert-bleu (cyan)
7	Bleu clair	15	Blanc

COLOR

Avant de dessiner sur l'écran, il vous faut indiquer à l'ordinateur dans quelle couleur vous voulez travailler :

```
]COLOR=2
```


Bien entendu, si votre écran est en noir et blanc, vous n'obtiendrez pas ces couleurs, mais des gris de diverses intensités.



La grille graphique en basse résolution

Si vous omettez d'indiquer la couleur en cours de dessin, le Basic se référera à la dernière couleur programmée. Mais il en faut obligatoirement une au départ.

PLOT

L'espace réservé au graphique (basse résolution) représente donc l'équivalent des 20 lignes supérieures de texte, mais cette fois divisé en 40 lignes et 40 colonnes numérotées de 0 à 39 (lignes et colonnes). Cette grille permet l'affichage de 1600 petits pavés rectangulaires, à volonté ; un pavé est spécifié par son numéro de ligne et son numéro de colonne, grâce à l'ordre spécial PLOT (« dessiner »). La syntaxe de PLOT est :

PLOT colonne, rangée

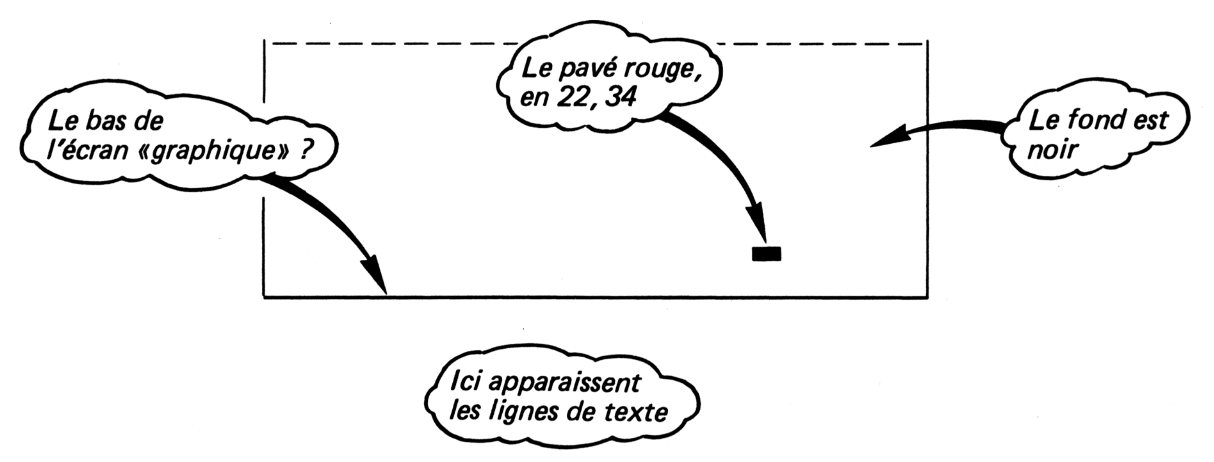
Vérifions-le en allumant un pavé rouge (couleur = 9) à l'intersection de la colonne 22 et de la rangée 34, donc vers le bas de l'écran.

Notez qu'en mode GR (graphique), les ordres s'inscrivent dans la zone du bas de l'écran, de 4 lignes, destinée au texte.

Frappez :

```
]COLOR=9
]PLOT 22,34
```

Au RETURN, vous voyez :



4. Quelques tracés graphiques

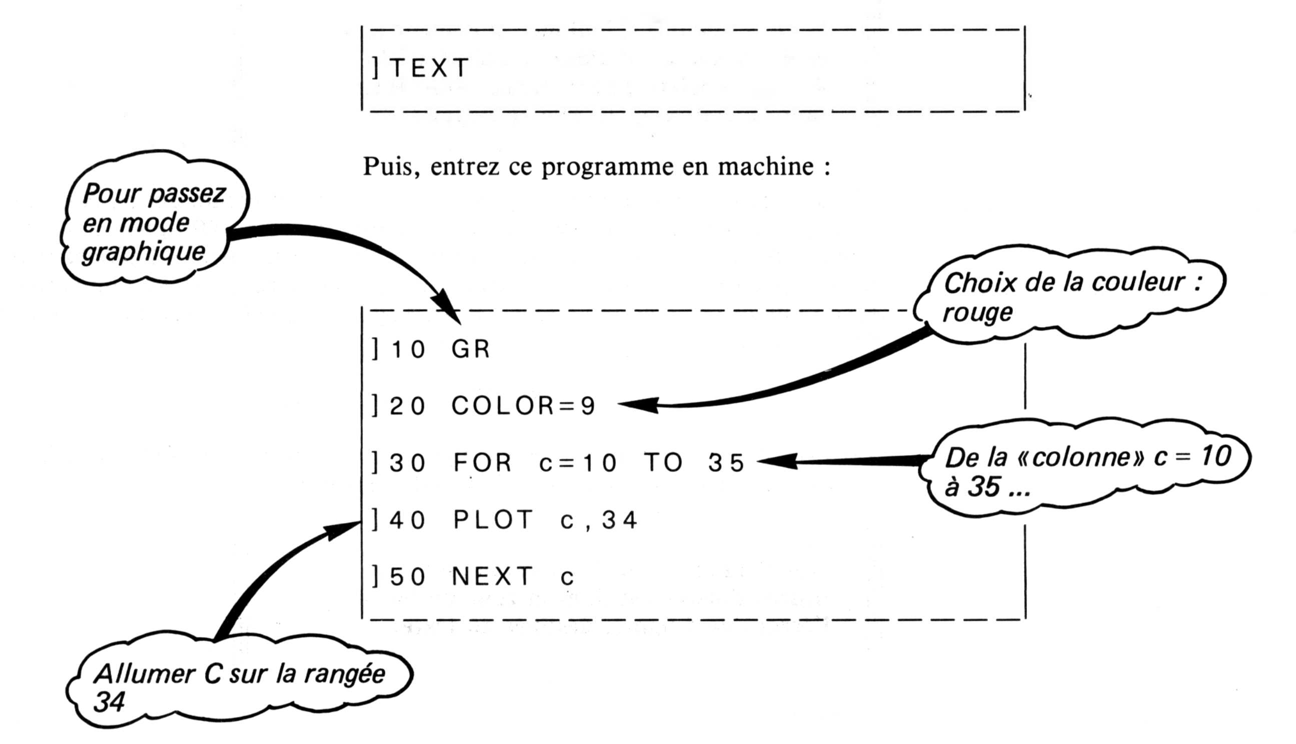
Voulez-vous tracer un trait rouge sur cette même rangée, de la colonne 10 à la colonne 35 ? Pour cela, on va créer un programme mais plutôt que de l'entrer en mode graphique dans cette minuscule « fenêtre » de 4 lignes, on va le frapper en mode texte puis le lancer.

La première instruction consistera à programmer le mode graphique basse résolution, soit GR.

Revenez au mode texte en frappant TEXT :

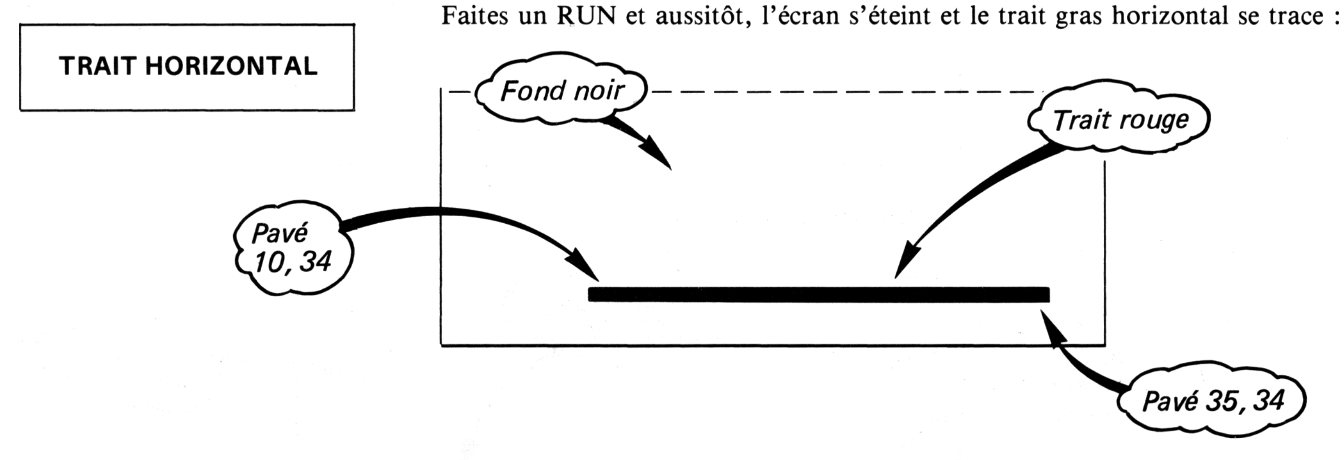
```
]TEXT
```

Puis, entrez ce programme en machine :

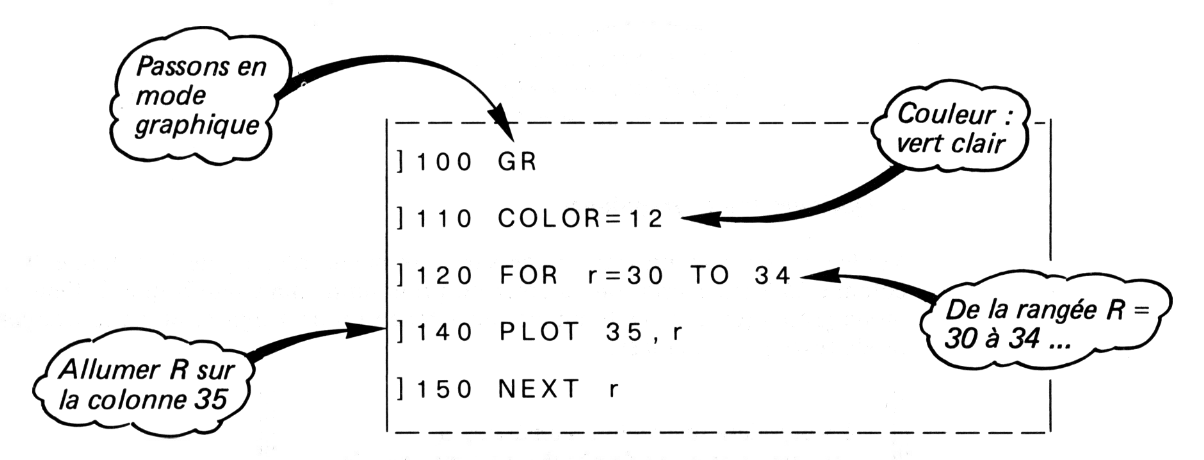


Faites un RUN et aussitôt, l'écran s'éteint et le trait gras horizontal se trace :

TRAIT HORIZONTAL



Voulez-vous un trait vertical, de 35,30 à 35,34 ? Ne touchez pas au programme précédent : on va en écrire un second qui commencera à la ligne 100. Repassez en mode TEXT, et entrez ce programme en machine :



Il faut maintenant le lancer ; or, on trouve *deux* programmes en mémoire, que faire ?

RUN N° DE LIGNE

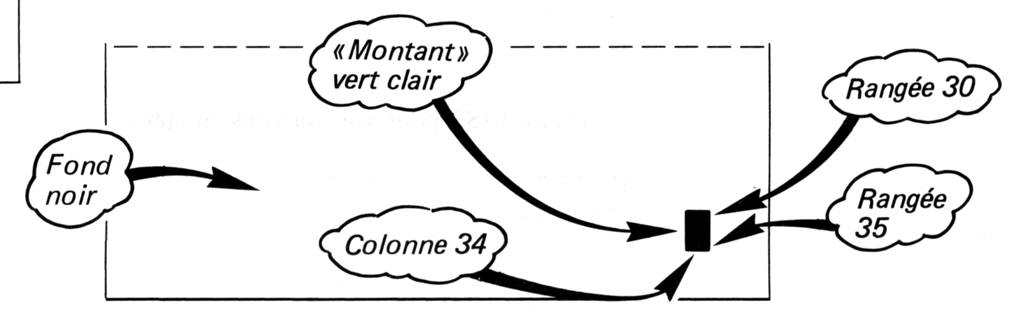
Pour lancer le programme commençant à une ligne donnée, indiquez le numéro de cette ligne après RUN.

Ainsi, lancez ce programme avec :

```
]RUN 100
```

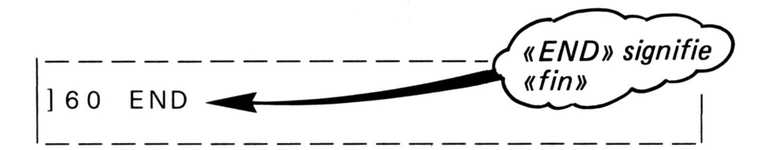
TRAIT VERTICAL

C'est bien notre second programme qui démarre ; il va faire apparaître sur l'écran un « montant » vertical :



Voulez-vous ne relancer *que* le progamme du trait horizontal ? Si vous faites RUN sans numéro de ligne, c'est bien lui qui va démarrer mais il sera suivi aussitôt par le programme de trait vertical qui effacera le tracé précédent. Il faut donc indiquer au Basic de s'arrêter, ce qu'on peut faire en plaçant, en 60, une nouvelle instruction :

END



Lancez le tout avec RUN tout court : seul le trait horizontal s'affiche car le Basic a buté sur END et s'est arrêté.
Pour faire apparaître à la fois le trait horizontal rouge et le montant vert, il faut supprimer à la fois ce END de la ligne 60 mais aussi le GR de la ligne 100 car il efface tout ce qui précédait sur l'écran. Faites-le en frappant 60 puis ↲, 100 puis ↲, et faites un LIST :

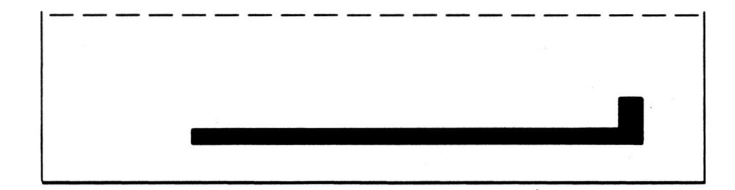
```
]60

]100

]LIST

   10 GR
   20 COLOR = 9
   30 FOR c = 10 TO 35
   40 PLOT c, 34
   50 NEXT c
  110 COLOR = 12
  120 FOR r = 30 TO 34
  140 PLOT 36, r
  150 NEXT r
```

Lancez ce programme avec RUN tout court et vous obtiendrez :



Pourquoi, maintenant, ne pas terminer ce rectangle ? Ajoutez simplement les *deux* instructions suivantes :

```
]45 PLOT c,30

]130 PLOT 10,r
```

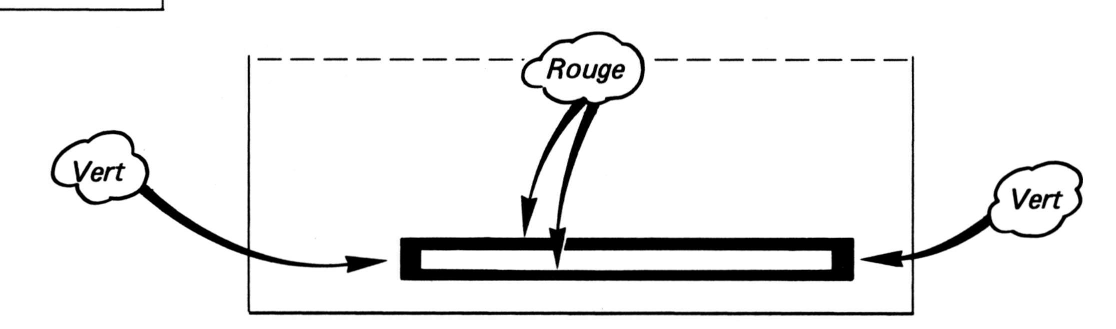
Faites un LIST pour voir où vous en êtes :

```
]LIST

  10 GR
  20 COLOR = 9
  30 FOR c = 10 TO 35
  40 PLOT c, 34
  45 PLOT c, 30
  50 NEXT c
 110 COLOR = 12
 120 FOR r = 30 TO 34
 130 PLOT 10, r
 140 PLOT 36, r
 150 NEXT r
```

RECTANGLE

Lancez tout cela avec un RUN tout court afin d'obtenir :



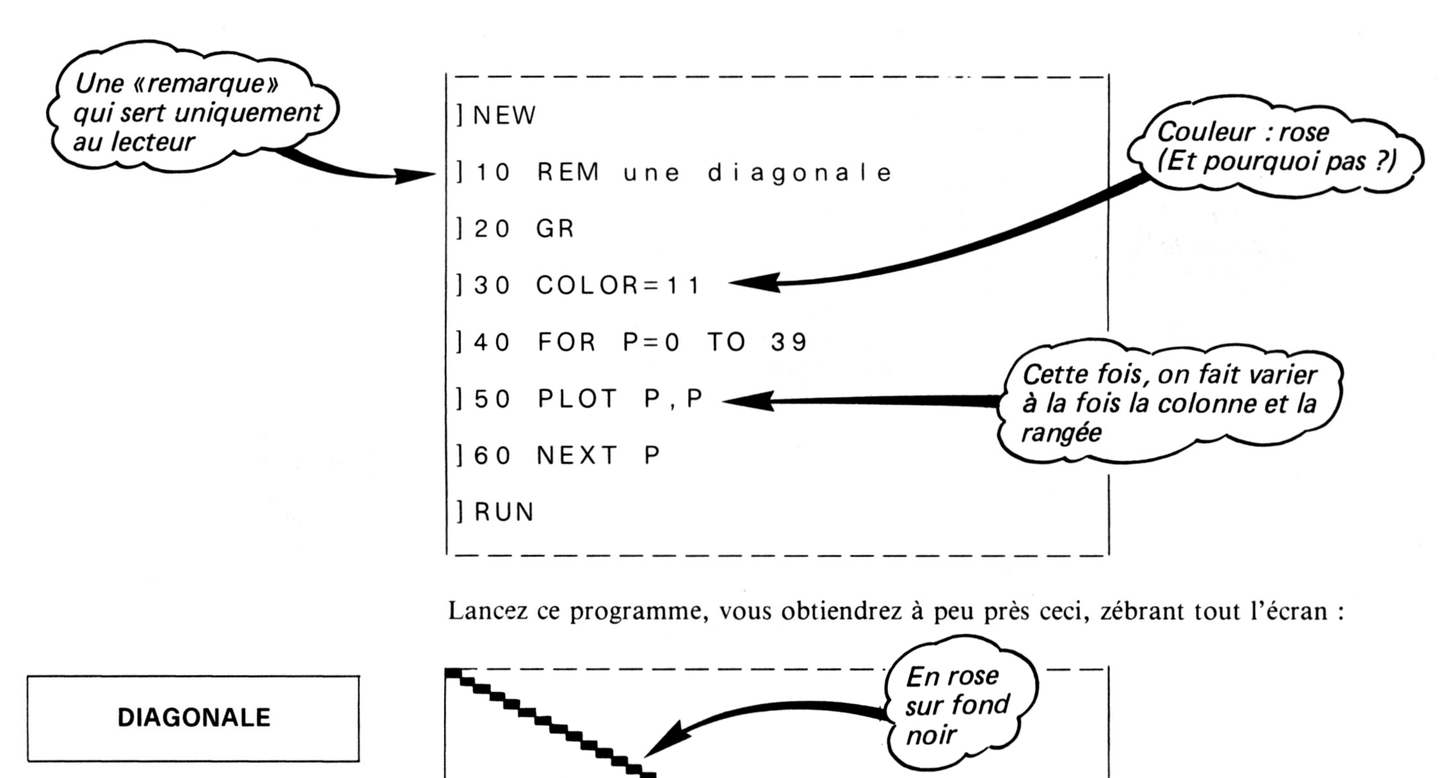
Etablissons maintenant une diagonale (en escalier) ; effacez le programme avec un NEW, puis entrez ces nouvelles instructions. On les a fait précéder d'une sorte de titre.

REMARQUES

REM

On a le droit d'ajouter des réflexions personnelles sur un programme, chargées de le commenter. Ces lignes doivent être précédées de la commande REM, pour « remarque ».

Lorsque le Basic rencontre la commande REM, il saute tout ce qui la suit puis passe à l'instruction suivante. Plaçons donc une REM en 10, ce qui donne :

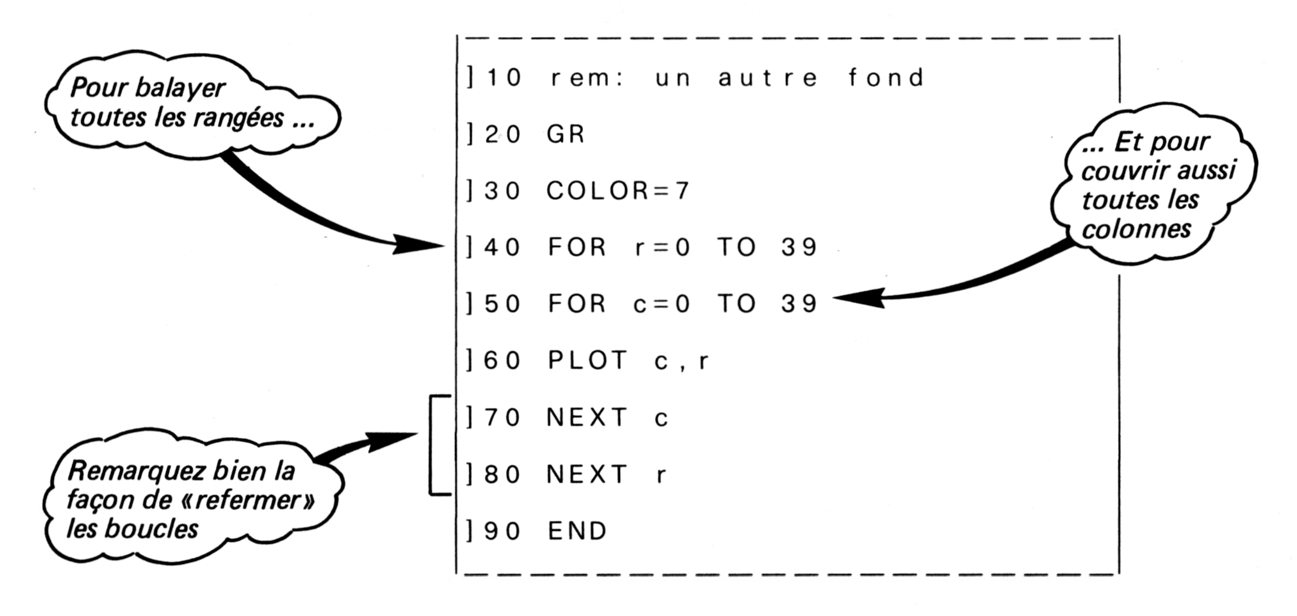


Lancez ce programme, vous obtiendrez à peu près ceci, zébrant tout l'écran :

DIAGONALE

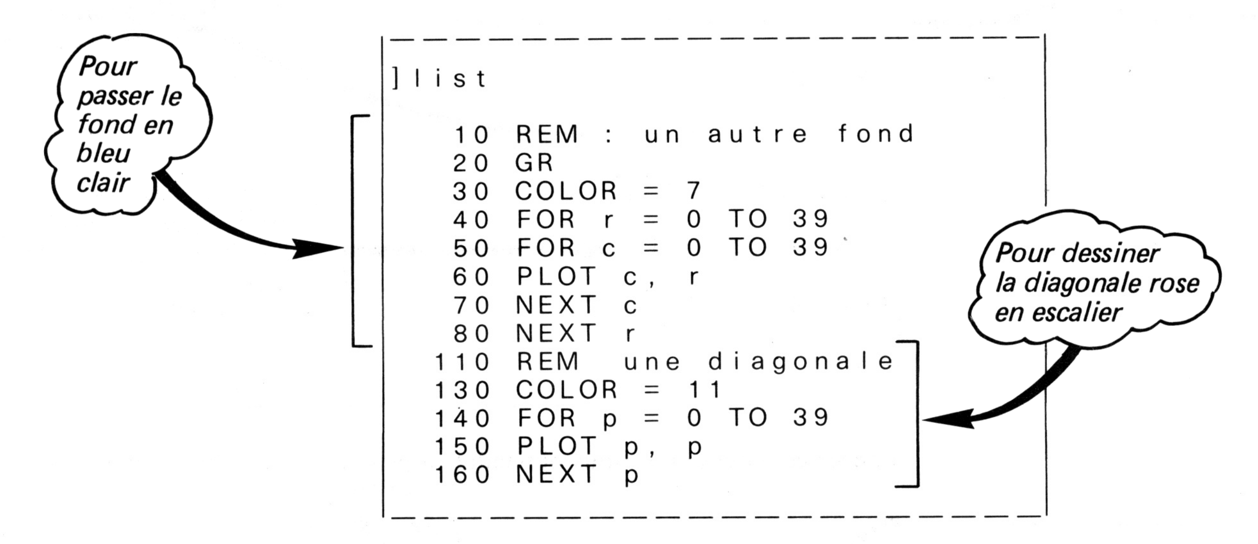
5. Pour mettre le fond en couleur

Peut-être ce fond noir vous déplaît-il ? Qu'à cela ne tienne, passons-le en bleu clair (couleur 7) :



FOND COULEUR

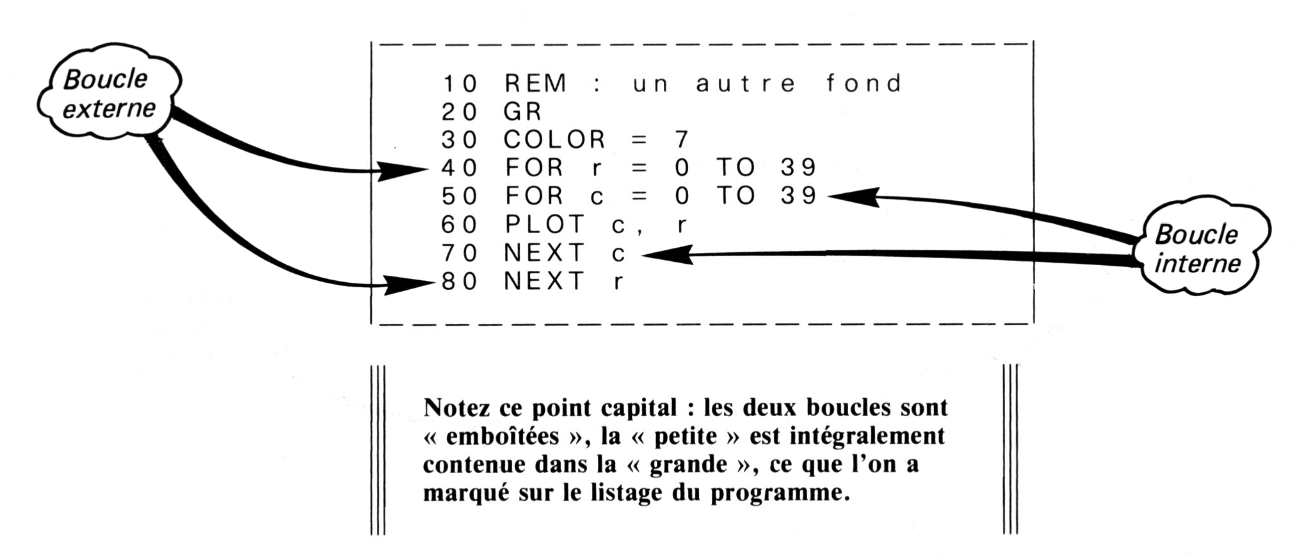
Ce programme est intéressant parce qu'il comporte *deux* boucles FOR... TO à NEXT, aussi va-t-on y revenir. Pour l'instant, lancez-le : vous allez voir enfin l'écran graphique dans sa totalité, en bleu clair.
Voulez-vous ajouter la diagonale rose sur ce fond bleu ? Ajoutez la même série d'instructions que précédemment, ce qui donne :



Faites un RUN afin de constater que l'effet est, somme toute, plutôt heureux.

6. Règles avec les boucles

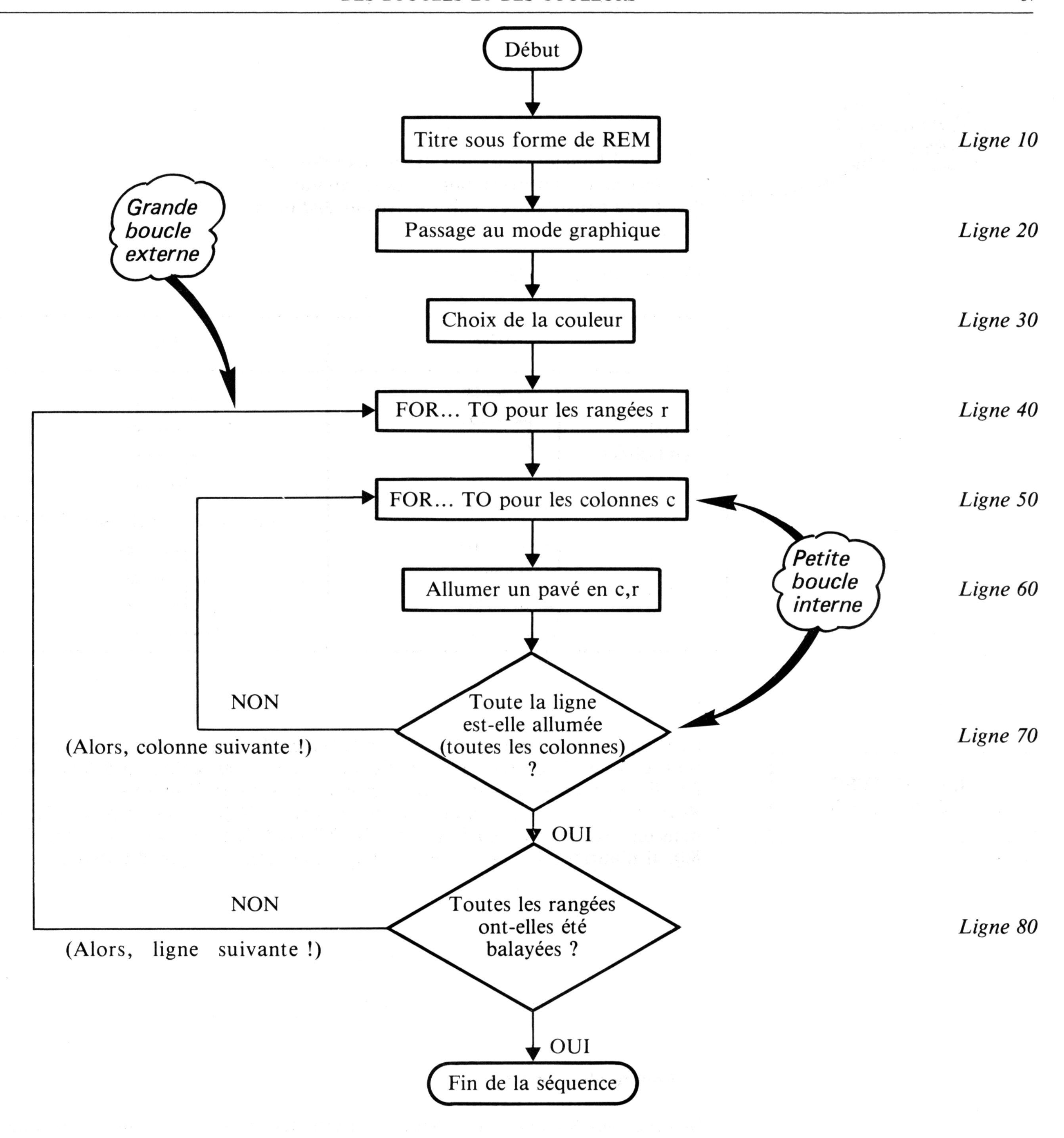
Reprenons le programme précédent, qui passait le fond en bleu ; voici son listage, avec des flèches mettant en évidence la présence des boucles :



Notez ce point capital : les deux boucles sont « emboîtées », la « petite » est intégralement contenue dans la « grande », ce que l'on a marqué sur le listage du programme.

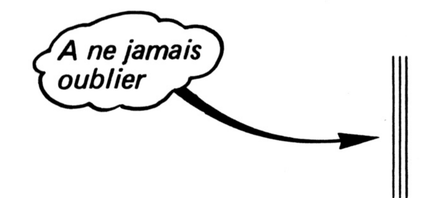
EMBOITAGE DES BOUCLES

L'organigramme du programme montre bien, lui aussi, la présence de ces deux boucles :



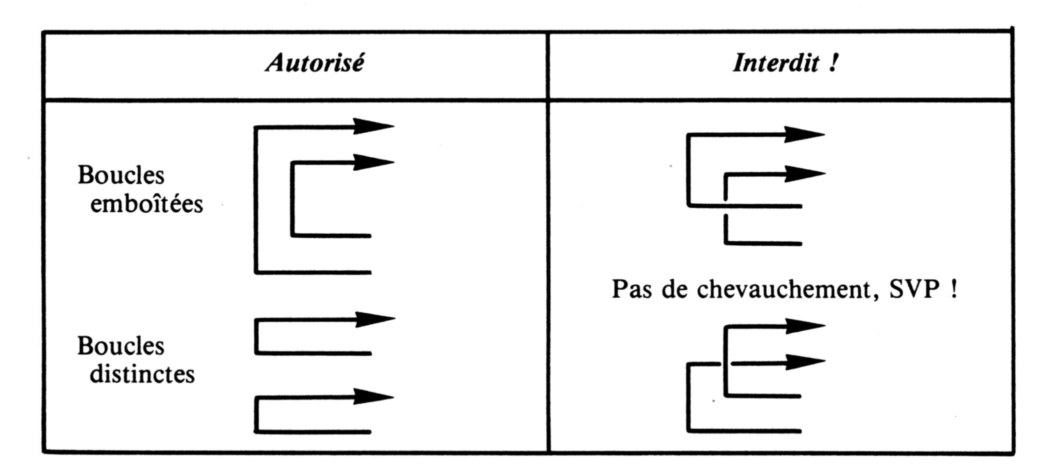
Organigramme des deux boucles emboîtées

Remarquez par quoi se traduisent les lignes NEXT, dans cet organigramme : par des questions. Chacune peut obtenir l'une des deux réponses OUI ou NON, d'où le dessin d'un losange pour les indiquer avec *deux* sorties possibles. Notez ici ce point fondamental :



Deux boucles (deux, ou davantage) doivent être soit totalement emboîtées (contenues l'une dans l'autre), soit totalement distinctes.

On peut illustrer cela ainsi :



REGLES AVEC LES BOUCLES

Cela signifie encore que, si on ouvre la boucle R, puis on démarre la boucle C, *il faut* refermer d'abord C (avec un NEXT C) puis *après* seulement R (avec un NEXT R). Sinon, vous commettriez cette erreur typique des débutants.
Revoyez le programme ci-dessus : on a *ouvert* (FOR... TO) la boucle R, puis C, mais on a refermé d'*abord c* (avec le NEXT de la ligne 70) puis R (avec le NEXT de 80). Il n'aurait pas fallu faire l'inverse : attention à ce type d'erreur !

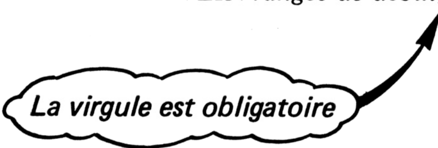
7. Encore plus simple !

HLIN
VLIN

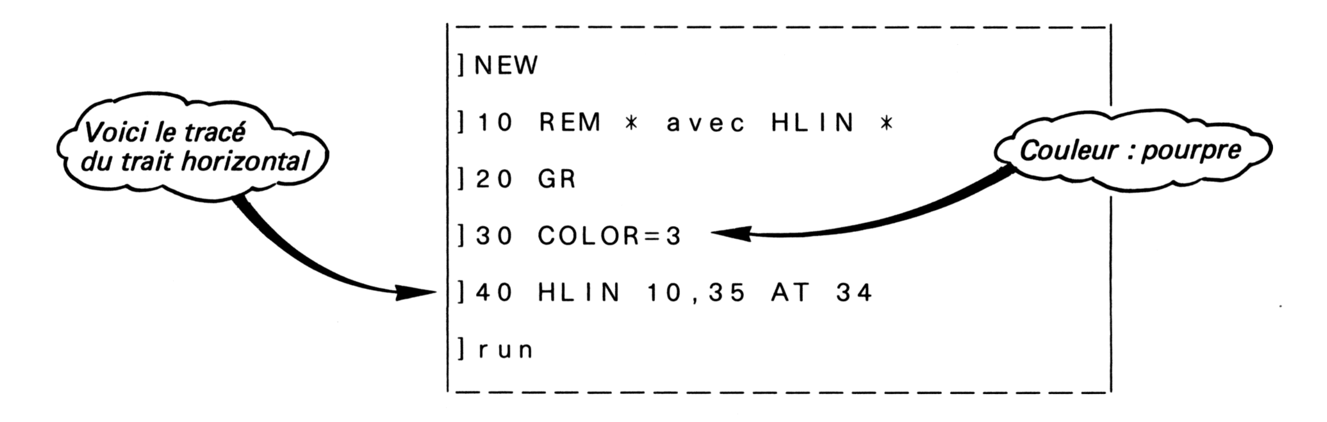
Si vous avez bien compris comment tracer des droites, nous allons vous montrer comment on peut supprimer les boucles FOR... TO et simplifier le programme. Pour cela, on va employer deux nouvelles instructions, HLIN qui trace une *ligne horizontale* et VLIN qui trace une ligne verticale. La syntaxe de ces ordres est :

HLIN *colonne de début,colonne de fin* AT *ligne*

VLIN *rangée de début,rangée de fin* AT *colonne*

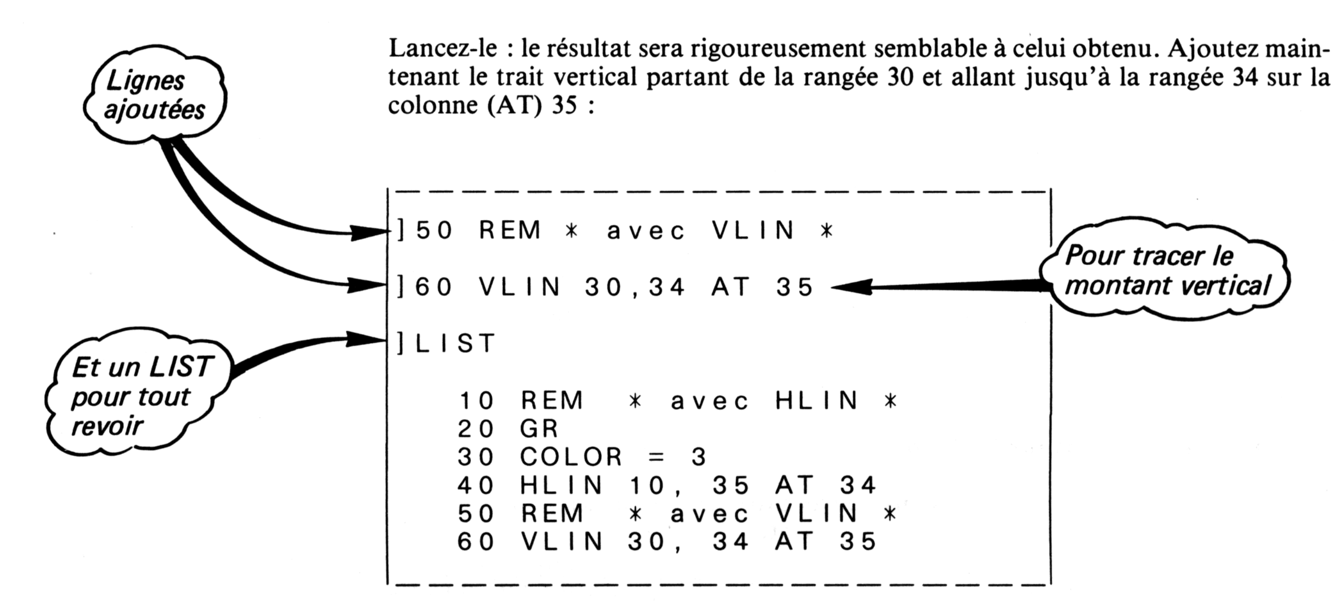


Remplacez simplement les trois données en italique par leur valeur, d'après la grille graphique, et vous pourrez tracer des droites. Expérimentons-le en traçant *les mêmes droites* que dans nos programmes précédents. L'horizontale de 10 à 35 « sur la rangée » (ce que l'on note AT) 34 :




```
]NEW
]10 REM * avec HLIN *
]20 GR
]30 COLOR=3
]40 HLIN 10,35 AT 34
]run
```

Lancez-le : le résultat sera rigoureusement semblable à celui obtenu. Ajoutez maintenant le trait vertical partant de la rangée 30 et allant jusqu'à la rangée 34 sur la colonne (AT) 35 :




```
]50 REM * avec VLIN *
]60 VLIN 30,34 AT 35
]LIST

   10 REM  * avec HLIN *
   20 GR
   30 COLOR = 3
   40 HLIN 10, 35 AT 34
   50 REM  * avec VLIN *
   60 VLIN 30, 34 AT 35
```

Lancez ce programme : vous obtiendrez le même dessin que précédemment, mais dans la même couleur, ici, car on n'en a pas changé en cours de route. N'est-ce pas plus simple ici ? Oui, mais ce principe ne vous permettra pas de créer un pointillé, horizontal par exemple ; voici comment le faire avec un FOR... TO :

```
   10 REM  - un pointillé -
   20 GR
   30 COLOR = 3
   40 FOR c = 0 TO 39 STEP 2
   50 PLOT c, 34
   60 NEXT c
```

POINTILLE

Lancez ce programme et en effet, vous allez voir ceci :



Ainsi, on a tracé le pointillé demandé.

Réponses aux questions sur le chapitre IV

1. Non, il n'y a pas d'erreur, ici. Vous pouvez attribuer aux lignes n'importe quel numéro : respectez seulement un ordre croissant.

2. L'exécution d'un programme est commandée par RUN. L'ordre GO n'existe pas, tel, de toute façon (sauf sous des formes telles que GOTO).

3. Le point-virgule, dans un ordre PRINT, supprime le retour à la ligne. L'affichage se poursuivra sur la même ligne.

4. Oui, l'ordre LIST ré-affiche le programme en mémoire.

5. Pour supprimer une ligne de programme, il faut frapper son numéro puis ↵. Rappelez-vous que CONTROLE/X sert à abandonner ce qu'on vient de frapper ; la machine n'en tiendra pas compte ; par conséquent, il n'y aurait pas suppression.

6. NEW sert à effacer la mémoire (vive, centrale), qui contient le programme, les données et les résultats.

7. Ces trois ordres effacent l'écran ; à vous de choisir celui que vous préférez ! Rappelez-vous que la frappe de CONTROLE/L ne peut se faire qu'en mode direct. Par contre, HOME et ?CHR$(2) peuvent intervenir en mode direct ou en mode programmé. On préfèrera HOME, tout simplement parce que c'est plus court à frapper et plus évident.

8. C'est INPUT qui sert à introduire de nouvelles données en cours de l'exécution d'un programme (on a inventé, ici, le mot ENTER pour les besoins de la question).

9. Ici, on a répété deux fois la même chose, d'abord de façon « savante », puis très simplement. En effet, un « algorithme » est la « logique » du programme, la façon dont il est organisé pour atteindre le résultat voulu. Et « représentation graphique » ou « dessin » sont très synonymes.

1Ø. Pour interrompre un programme, il faut faire CONTROLE/C.

Questions sur le chapitre V

N'hésitez pas, dès maintenant, à imaginer par vous-même des courtes séquences de programmes. C'est, de loin, le meilleur exercice. Mais voyons, au préalable, si vous répondrez facilement à ces quelques questions fondamentales.

Y a-t-il ou n'y a-t-il pas d'erreur dans les lignes suivantes extraites de leur contexte ?

1. FOR N=1 TO N=23
2. FOR N=37 TO 73
3. FOR N=37 TO 73 STEP 25
4. FOR N=37 TO 73 STEP 100
5. FOR N=37 TO 73 STEP .01
6. FOR N=37 TO 73 STEP −25
7. FOR N=37 TO 0 STEP −.1
8. FOR N=1 TO 3:?N: NEXT N
9. FOR N=T+2 TO T+5:?T: NEXT T

Bon ! Si vous vous en êtes bien sorti, bravo. Continuons alors :

10. Pour passer en mode graphique basse résolution, il faut faire :
    - ☐ GR
    - ☐ TEXT
11. Pour allumer un point graphique, on commandera :
    - ☐ PRINT
    - ☐ PLOT
12. Peut-on lancer un programme à partir de n'importe quelle ligne ?
    Oui
    Non

Réponses page 92

Réponses aux questions sur le chapitre V

1. Non, ça ne va pas ! Le nom de la variable ne doit être écrit qu'*une seule fois*. Il fallait écrire :

FOR N = 1 TO 23

2. Cette ligne est parfaitement correcte.

3. Celle-ci aussi.

4. Oui, celle-ci aussi : le pas est très grand, aussi n'y a-t-il que le 37 qui sera pris en compte car, au tour suivant, on passerait à 137 ce qui excède la limite de 73.

5. C'est parfaitement correct : le pas peut être tout petit.

6. Alors là, rien ne va plus : essayez donc de *décompter* (−25) en accroissant des valeurs de 37 à 73 !

7. Ici, tout va bien car, avec le pas négatif, on a inscrit un décomptage de 37 à zéro.

8. C'est parfaitement correct. Rappelez-vous que le double point est un séparateur d'ordres sur une même ligne.

9. Mais non, il y a une confusion entre la variable N et T qui sert à la définir. Il aurait fallu écrire, apparemment :

FOR N = T + 2 TO T + 5:?N: NEXT N

10. GR fait passer en graphique basse résolution (alors que TEXT fait revenir au mode habituel).

11. La mise en service d'un point graphique se fait avec PLOT.

12. Oui, on peut lancer un programme à partir de n'importe quelle ligne : il suffit de l'indiquer dans le RUN. Par exemple, RUN 367 fera démarrer le programme à la ligne 367 (encore doit-elle exister).

CHAPITRE VI

JEUX DE HASARD

Grâce au tirage de nombres aléatoires, on va pouvoir simuler le jeu de loto ou de 421, et créer une mosaïque multicolore. Mais on va aussi étudier bien des choses, ici, l'ordre de branchement conditionnel IF... THEN en tout premier lieu. On finira par un programme d'enseignement assisté par ordinateur.

Principaux thèmes étudiés

Nombres aléatoires	Branchement conditionnel
RND	Minuteur
Notation scientifique	Opérateurs relationnels
Entiers	Vrai
INT	Faux
Virgule dans un PRINT	Relations de chaînes
Tabulation	ASCII
Loto	Comparaison de chaînes
TAB	NOT
PRINT TAB	AND
Jeu de 421	OR
Mosaïque	Opérateurs logiques
Temporisation	Enseignement assisté
IF... THEN	EAO

1. Nombres au hasard

NOMBRES ALEATOIRES

RND

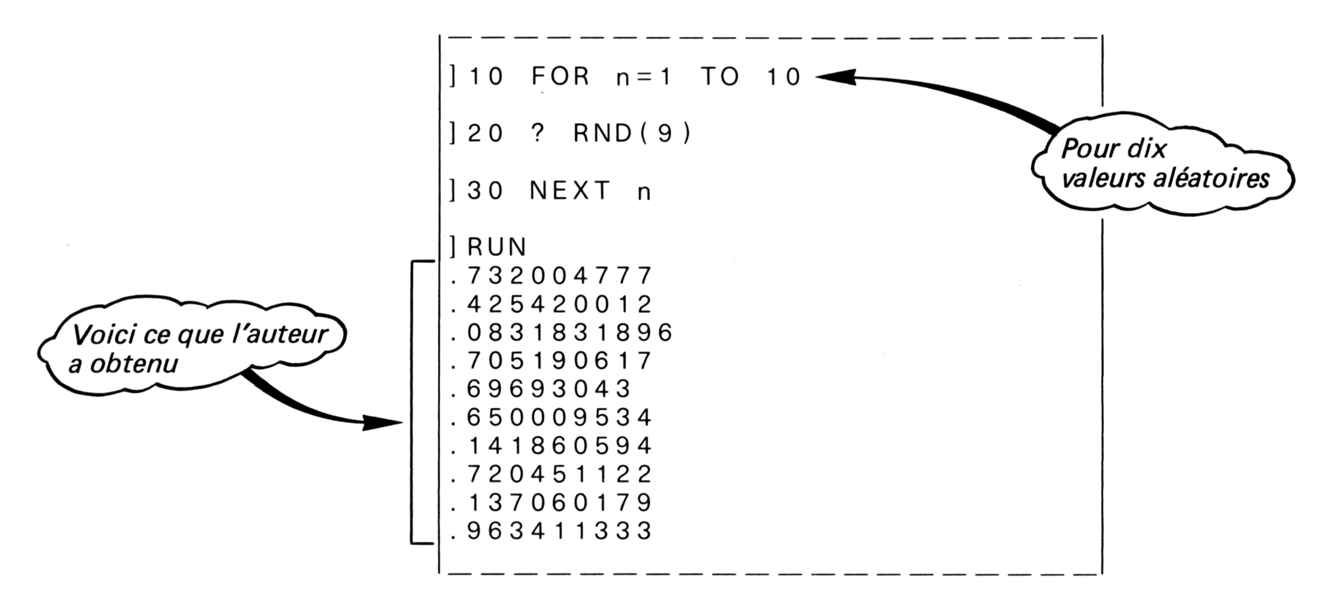
Une fonction spéciale du Basic sert à tirer des nombres au hasard, c'est RND ; ces lettres proviennent du mot anglais « *random* », soit *aléatoire,* au hasard. L'ordre RND demande à être suivi par un nombre *absolument quelconque* mais placé entre parenthèses ; le Basic a, parfois, de telles fantaisies... Ne mettez ni zéro, ni nombre négatif, et par paresse, utilisez le 9 qui se trouve situé sur la même touche que l'ouverture des parenthèses. Essayez simplement en mode direct :

```
]? RND(9)
.732004777
```

Ordre de tirage et d'affichage d'un nombre aléatoire

L'auteur a obtenu cette valeur bizarre, un peu inférieure à 1. Vous tirerez peut-être une autre valeur

Créons un court programme pour afficher plusieurs valeurs aléatoires. Voici ce programme, suivi d'un RUN :



```
]10 FOR n=1 TO 10
]20 ? RND(9)
]30 NEXT n
]RUN
.732004777
.425420012
.0831831896
.705190617
.69693043
.650009534
.141860594
.720451122
.137060179
.963411333
```


Il est probable que vous obtiendrez des valeurs différentes ; en tous cas et à chaque remise en service de l'ordinateur, vous retrouverez la même série.

En effet, ces valeurs sont appelées « aléatoires » mais elles sont en réalité « pseudo-aléatoires ». Elles appartiennent à une très grande série répétitive de nombres, toujours inférieurs à 1.

NOTATION SCIENTIFIQUE

Parfois même, vous serez amené à travailler avec des nombres tellement petits qu'ils seront affichés en *notation scientifique.* Ce type de notation utilise les puissances ; si vous les avez étudiées, cela ne vous déconcertera pas. Ainsi, rappelez-vous qu'un 1 suivi de douze zéros (soit 1000000000000) peut s'écrire aussi 10^{12} en écriture courante. De même, un millionnième (0,000001) peut s'écrire 10^{-6}. En informatique, on écrira :

1E + 12 pour 10^{12}

1E – 06 pour 10^{-6}

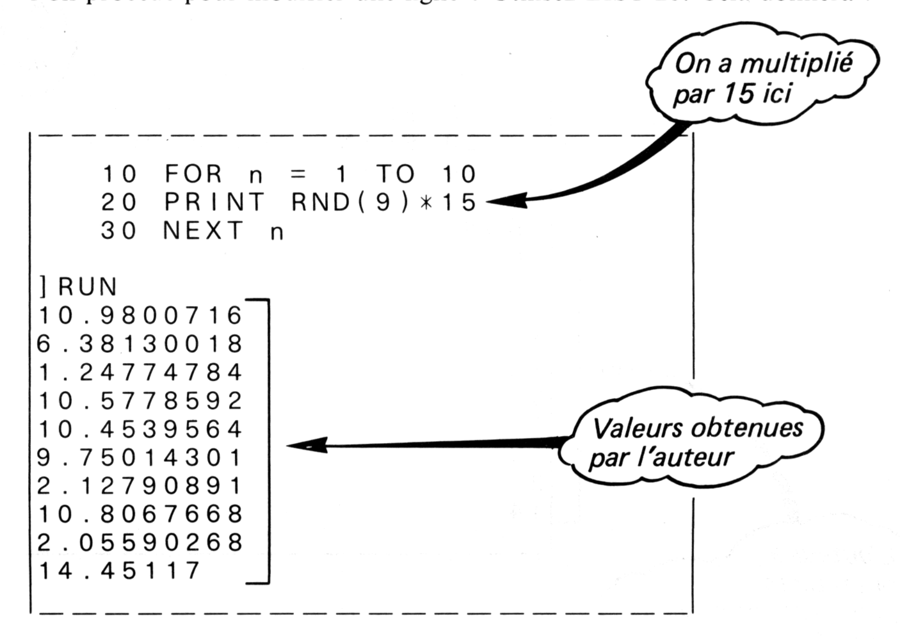
La lettre E représente la « base » 10 ; le cœfficient qui suit, avec son signe, est l'*exposant.* Avant le E peuvent se trouver d'autres chiffres, avec le signe *plus* ou *moins.*

C'est ce mode d'écriture qu'on appelle donc la *notation scientifique ;* ne vous laissez pas démonter par elle et, à la rigueur, ignorez les valeurs aléatoires en notation scientifique !

Mais revenons à nos tirages : les nombres sont bien petits, et il va falloir les multiplier sérieusement pour obtenir des valeurs supérieures.

D'abord, notez que les tirages sont compris entre 0 (qui peut être tiré) et 1 (mais le 1 ne sera jamais tiré : on ira jusqu'à 0.999999999...).

Si l'on veut obtenir des valeurs comprises entre 0 et 15, par exemple, il faudra multiplier les nombres tirés par 15. Modifiez la ligne 20 du programme précédent pour introduire la multiplication par 15, et relancez-le. Vous souvenez-vous comment l'on procède pour modifier une ligne ? Utilisez LIST 20. Cela donnera :




```
   10 FOR n = 1 TO 10
   20 PRINT RND(9)*15
   30 NEXT n

]RUN
10.9800716
6.38130018
1.24774784
10.5778592
10.4539564
9.75014301
2.12790891
10.8067668
2.05590268
14.45117
```

Remarquez qu'en multipliant par 15 des valeurs comprises entre 0 (inclus) et 1 (exclu), on peut obtenir toutes les valeurs entre 0 (inclus) et 15 (exclu), donc 14,9999...

Ces nombres décimaux sont encore très embarassants. On va éliminer la partie décimale (à droite de la virgule, pardon : du point décimal) en réclamant au Basic la seule partie *entière* de ces nombres. Sachant qu'*entier* se dit « *integer* » en anglais, vous comprendrez pourquoi l'ordre se code INT.

ENTIERS
INT

Vérifiez le fonctionnement de INT en mode direct ; il exige que le nombre soit mis entre parenthèses à sa suite. Par exemple, faites :

```
] ? INT(123.456)
123
```

L'entier de 123,456, est bien 123

En passant, notez une singularité de l'ordre INT : il n'arrondit pas et *fournit toujours la valeur entière immédiatement inférieure.* Par exemple (vérifiez-le) :

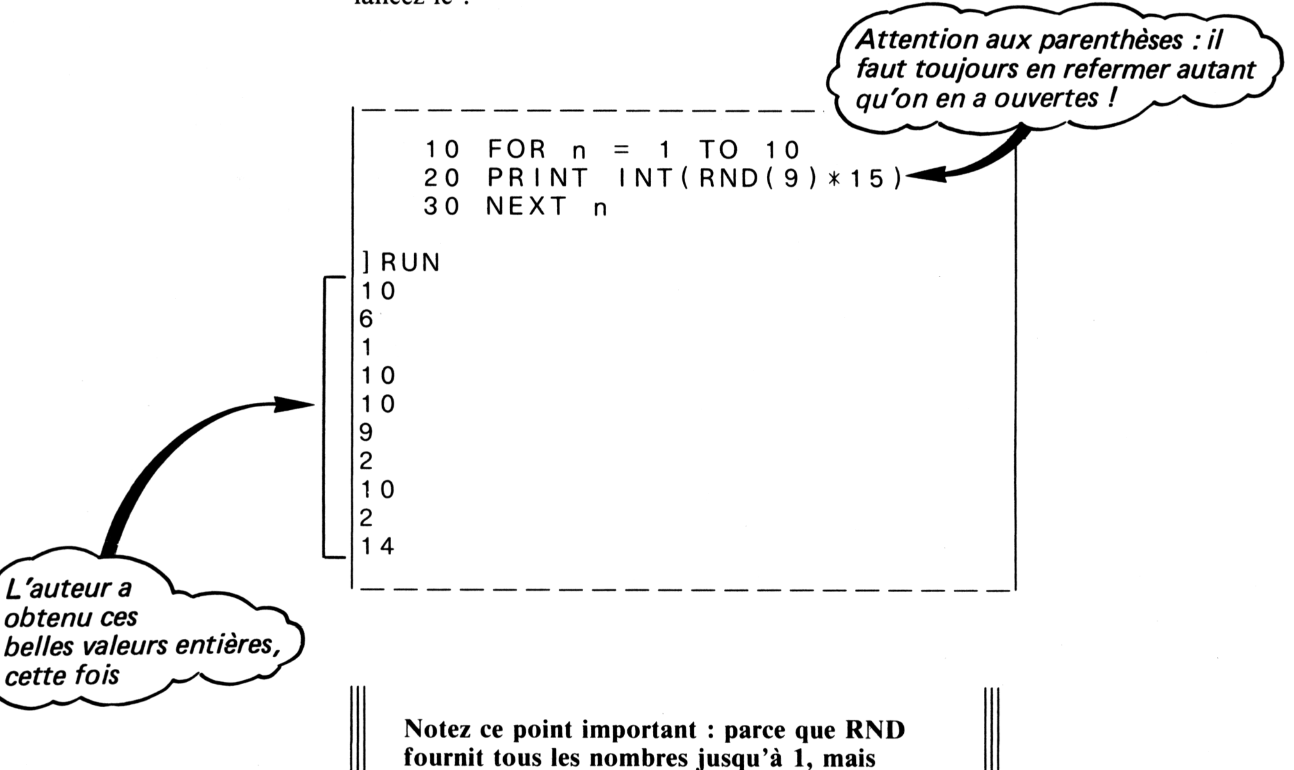
```
] ? INT(7.89)
7

] ? INT(-7.89)
-8
```

L'entier (immédiatement inférieur) de 7,89 est bien 7 ...

... Mais celui de –7,89 est –8

Refermons cette parenthèse pour en ouvrir d'autres, à la ligne 2Ø et comme annoncé. Modifiez cette ligne de façon que le programme soit le suivant, puis lancez-le :




```
     10 FOR n = 1 TO 10
     20 PRINT  INT(RND(9)*15)
     30 NEXT n

]RUN
10
6
1
10
10
9
2
10
2
14
```

Notez ce point important : parce que RND fournit tous les nombres jusqu'à 1, mais *jamais* 1, RND(9) * 15 donnera des valeurs jusqu'à 15, mais *jamais* 15.

Par conséquent, INT(RND(9) * 15) donnera toutes les valeurs entières de 0 (inclus) à 15 (exclu), donc de 0 à 14 *inclus*. C'est un peu compliqué, mais en vous y arrêtant un instant, vous comprendrez. Dès lors, nous voici près à attaquer les jeux.

2. Loto et 421

Avant de passer à ces deux jeux, notons encore le rôle d'un signe de ponctuation important, la *virgule*. En SmartBasic, la virgule commande une tabulation sur deux colonnes, à l'écran, lorsqu'elle est introduite dans un ordre PRINT. Vérifions-le :

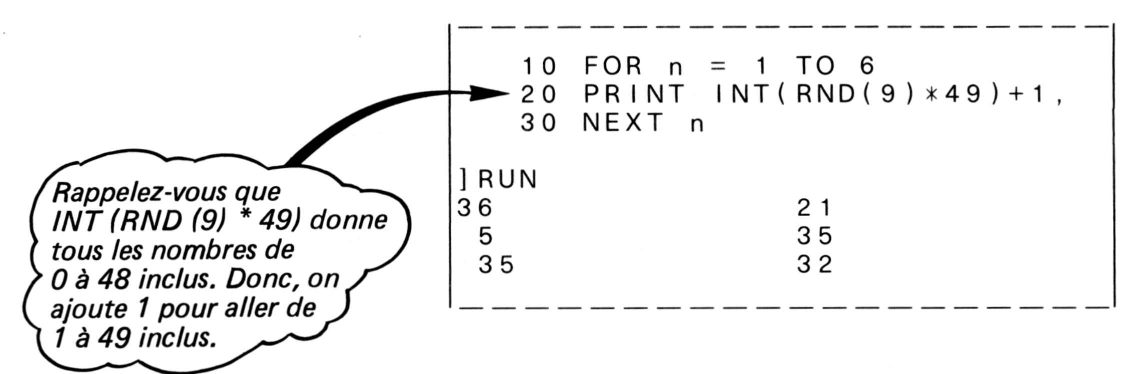
VIRGULE DANS UN PRINT
TABULATION

```
]?  1,2,3,4,5,6,7
1                2
  3              4
  5              6
  7
```

On obtient deux colonnes de 16 et 15 caractères. Rappelez-vous qu'avec un point-virgule, tout s'écrit à la suite :

```
]?  1;2;3;4;5;6;7
1234567
```

Passons alors au loto ; pour y jouer, il faut remplir une grille avec des nombres de 1 à 49 inclus. Si l'on veut 6 nombres, il suffit de les demander à l'ordinateur. Pour obtenir 49, il faudrait multiplier par 50 le nombre aléatoire tiré puis en extraire l'entier mais malheureusement, on risque aussi, de cette façon, de trouver un 0 dont le jeu de loto ne veut pas. Pour l'éviter, on ajoutera 1 au résultat, mais, pour ne pas aller jusqu'à 50, on va utiliser 49 et non 50 comme multiplicateur :




```
    10  FOR n = 1 TO 6
    20  PRINT  INT(RND(9)*49)+1,
    30  NEXT n

]RUN
36                    21
 5                    35
 35                   32
```

Sachez aussi que l'ordinateur ne vous procurera pas davantage de chances de gagner au vrai loto !
Faites maintenant un second RUN : surprise, vous retrouvez les mêmes valeurs ! C'est qu'en effet, chaque RUN relance *la même série* de nombres aléatoires ! Pour du hasard, ce n'est guère réussi : il faudrait démarrer à chaque fois une autre série. Or, c'est possible : il suffit de demander à l'ordinateur un nombre aléatoire avec un *argument négatif* dans les parenthèses ; parce que chaque argument négatif engendre une série qui lui est propre (et donc répétitive), on va faire mieux : c'est vous qui lui en fournirez l'argument. Ajoutez ces trois instructions à votre programme :

```
]2  REM + Le jeu du loto +

]6  INPUT a

]8  X=RND(-a)
```

Notez bien que les lignes 6 et 8 ne vont servir qu'à « ré-ensemencer » la série mais que de toutes façons, deux entrées semblables en réponse au INPUT vous feront obtenir les mêmes valeurs ; voici le résultat, après un LIST :

LOTO

On ré-ensemence avec un 5

Et ici, avec un 6

```
     2  REM  + Le jeu du loto +
     6  INPUT a
     8  x = RND(-a)
    10  FOR n = 1 TO 6
    20  PRINT  INT(RND(9)*49)+1,
    30  NEXT n

]RUN
?5
20                   23
 15                  34
 31                  23

]RUN
?6
44                   11
 27                  3
 9                   19
]ETC, jusqu'à un CONTROL/C sui-
vi d'un RETURN
```



TAB

PRINT TAB

Selon une méthode semblable, on peut jouer à ce jeu de dés qui se pratique d'habitude sur le zinc des bistrots et s'appelle le 421. Le nombre d'or, gagnant, est justement 421, à obtenir en lançant 3 dés simultanément. Simuler 3 lancements de dés, vous devez maintenant savoir le faire ; aussi va-t-on en profiter pour examiner comment afficher ces trois lancements sur *une seule ligne d'écran,* en plaçant les trois résultats obtenus l'un à gauche, l'autre au centre, et le troisième à droite. C'est simple : il suffit de préciser dans l'ordre d'affichage PRINT sur quelle colonne cet affichage doit commencer, avec un ordre complémentaire de tabulation qui se dit TAB. Rappelez-vous que l'écran est sur 31 colonnes. La syntaxe de TAB est :

Parenthèses obligatoires

Point-virgule obligatoire

De 1 à 255 ; il y aura des retours à la ligne tous les 31 caractères

PRINT TAB*(colonne);texte*

Valeur numérique ou chaîne de caractères entre guillemets

Vérifions-le :

Le T de tabulation vient sur la 9ème colonne de la ligne en dessous

Ici, 180/31 donne 5 retours à la ligne ; le reste est 25, donc l'affichage commence en 25ème colonne et se poursuit sur la ligne suivante

```
]? TAB(9);"Tabulation à 9"
        Tabulation à 9

]? TAB(180);"Ici, on est à 180"



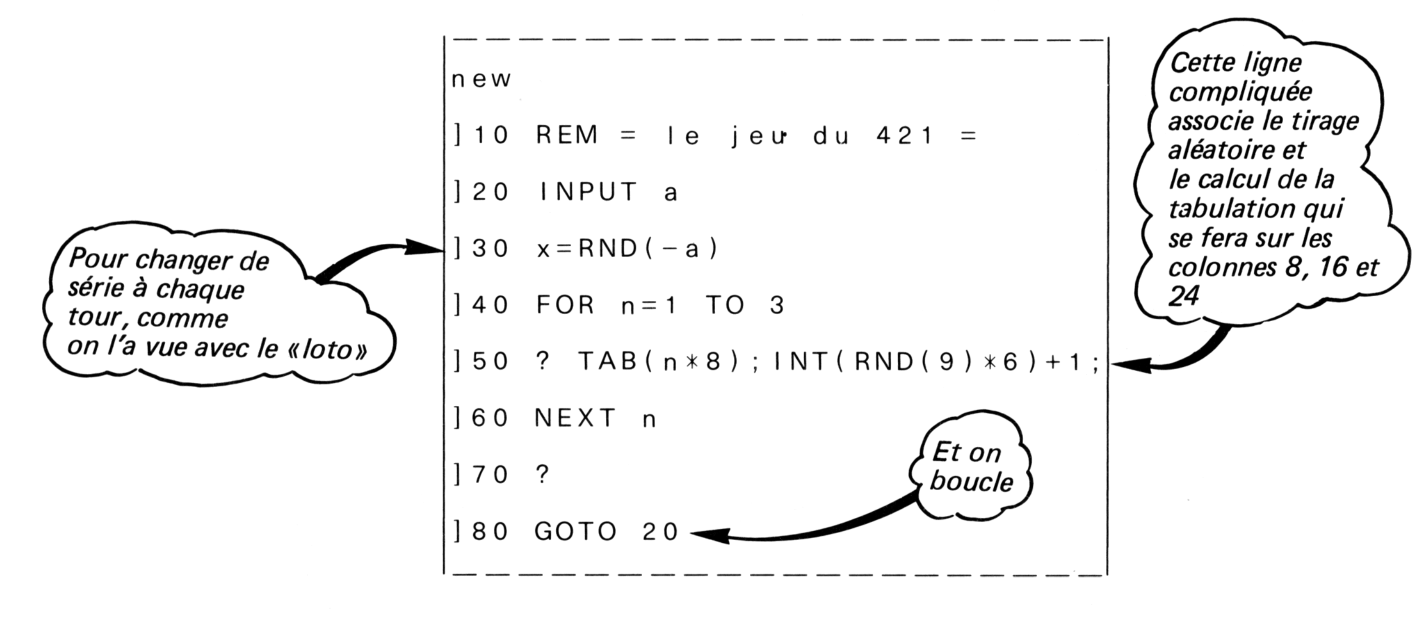
                        Ici, on
 est à 180

]? TAB(9);9;TAB(18);18
        9        18
```

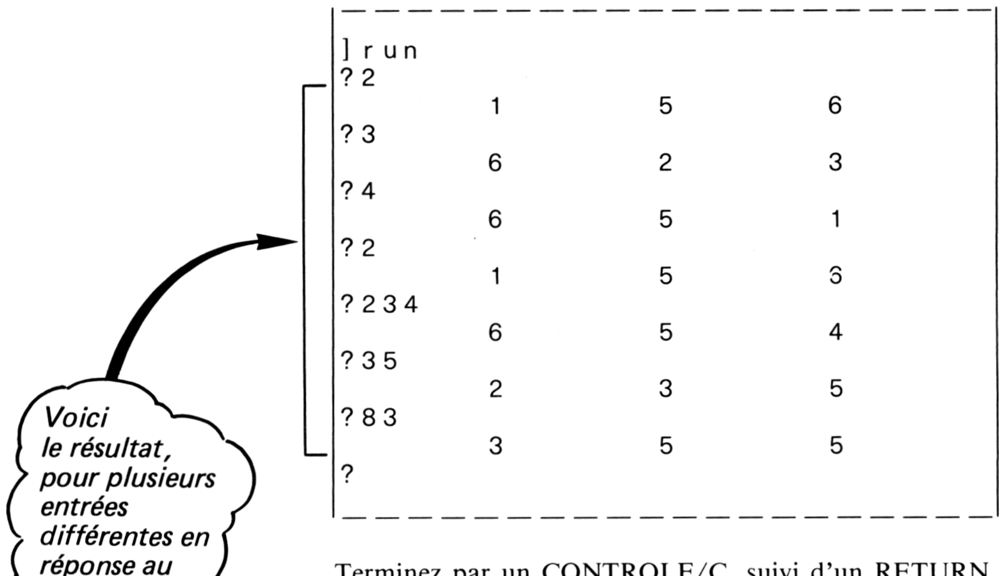
Enfin, on peut placer plusieurs TAB pour un unique PRINT

Notez bien que la tabulation commence toujours à partir du bord gauche de l'écran (et non à partir du caractère précédent).

Cela vu, créons un programme pour jouer au 421 en affichant les résultats en ligne. On va « boucler » ce programme avec un GOTO qui ramènera de la fin au début :



```
new
]10 REM = le jeu du 421 =
]20 INPUT a
]30 x=RND(-a)
]40 FOR n=1 TO 3
]50 ? TAB(n*8);INT(RND(9)*6)+1;
]60 NEXT n
]70 ?
]80 GOTO 20
```



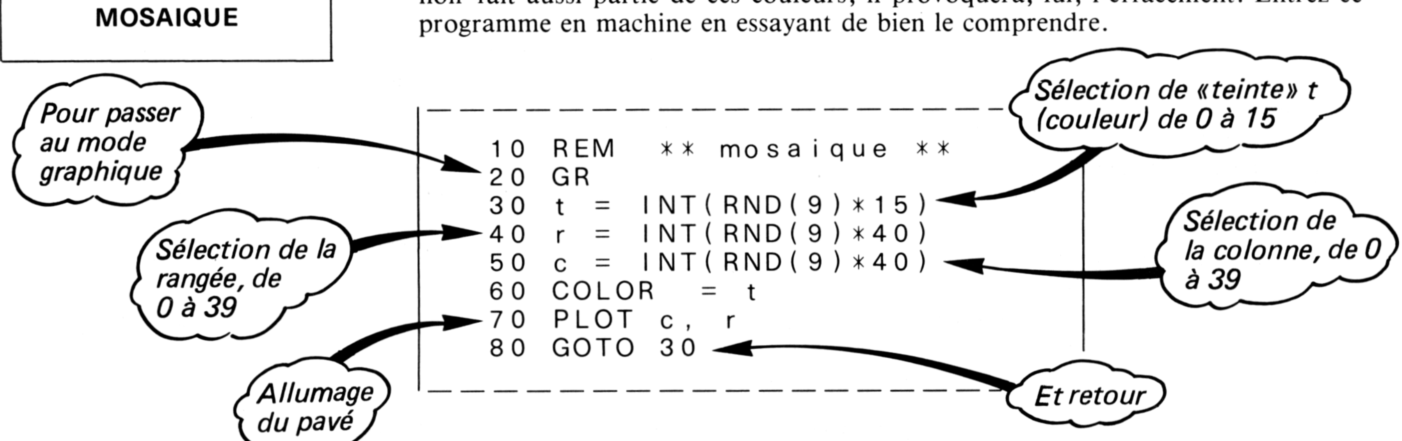
```
]run
?2
        1        5        6
?3
        6        2        3
?4
        6        5        1
?2
        1        5        6
?234
        6        5        4
?35
        2        3        5
?83
        3        5        5
?
```


Terminez par un CONTROLE/C, suivi d'un RETURN, et sachez encore que si l'argument de votre RND est un zéro, donc si vous faites RND(0), c'est la valeur *précédente* qui sera tirée.

3. Mosaïque

Puisque nous savons manipuler les couleurs, nous allons maintenant remplir l'écran *graphique* de pavés de couleur aléatoire placés aléatoirement. Puisque le noir fait aussi partie de ces couleurs, il provoquera, lui, l'effacement. Entrez ce programme en machine en essayant de bien le comprendre.

MOSAIQUE



```
10 REM  ** mosaique **
20 GR
30 t = INT(RND(9)*15)
40 r = INT(RND(9)*40)
50 c = INT(RND(9)*40)
60 COLOR = t
70 PLOT c, r
80 GOTO 30
```

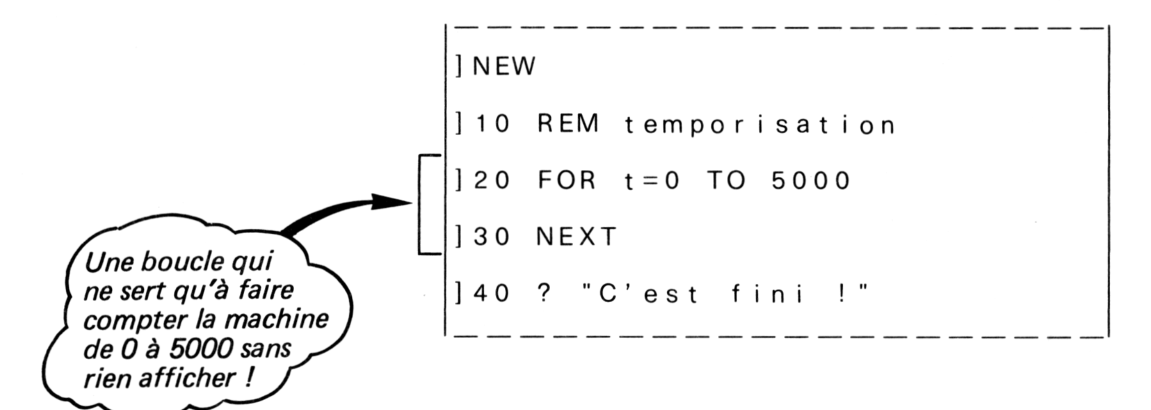

Lancez ce programme : vous verrez l'écran s'allumer et se recouvrir des pavés graphiques de toutes les couleurs, dans une mosaïque continuellement changeante. Pour le faire stopper, faites simplement CONTROLE/C, sans RETURN.

Avec CONTROLE/C, le ↲ n'est nécessaire que si l'arrêt a été provoqué au cours d'une demande d'entrée par INPUT.

4. Temporisation : comment battre la seconde

Une autre des fonctions que l'on peut accomplir et qui est d'une utilité certaine est la temporisation. Pour rapide qu'il soit, votre ordinateur ne travaille pas instantanément. Vous l'avez constaté avec la mosaïque précédente : les pavés graphiques se succédaient rapidement sur l'écran, mais avec une vitesse limitée toutefois. C'est qu'en effet, il faut à l'ordinateur « un certain temps » pour exécuter les instructions prescrites.

On va en profiter pour lui faire accomplir des choses parfaitement inutiles, sauf lui faire perdre du temps. Par exemple, et si vous avez un bon chronomètre, lancez le programme suivant, qui ne fait rien du tout, et chronométrez le temps qui s'écoule entre le RUN et l'affichage du message final :



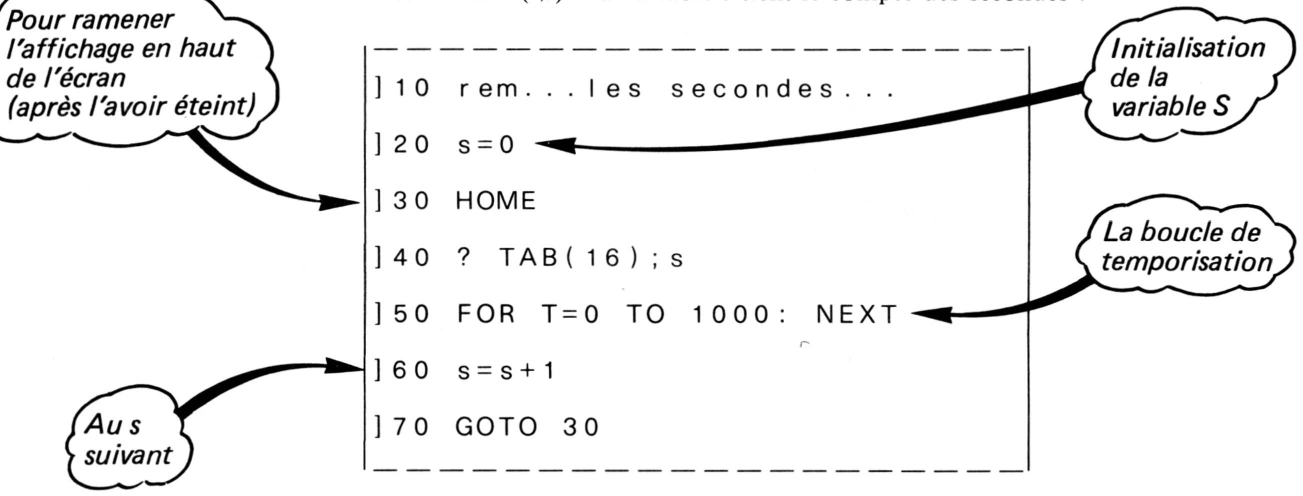

```
]NEW
]10 REM temporisation
]20 FOR t=0 TO 5000
]30 NEXT
]40 ? "C'est fini !"
```

TEMPORISATION

L'auteur a chronométré environ 5 secondes, ce qui signifie que Adam compte de 0 à 1000 en 1 seconde environ !

Remarquez également que dans ce programme, on n'a pas indiqué la variable après NEXT. Le Basic s'y retrouve parfaitement sans cette indication.

En effet, il aurait fallu, autrement, écrire NEXT T. Essayons maintenant de réaliser un compteur de secondes ; on affichera celles-ci en haut et au centre de l'écran, avec un TAB(16). La variable *s* tient le compte des secondes :




```
]10 rem...les secondes...
]20 s=0
]30 HOME
]40 ? TAB(16);s
]50 FOR T=0 TO 1000: NEXT
]60 s=s+1
]70 GOTO 30
```

Remarquez, en 60, la nouvelle affectation de *s* : on incrémente la variable de 1.

Notez aussi le rôle important de la ligne 20 : on fixe à la variable sa valeur initiale. On dit qu'on l'initialise.

Examinez aussi la boucle de temporisation : on a placé ses deux instructions sur une unique ligne en les séparant à l'aide d'un double point. Le résultat est le même que sur deux lignes ! Essayez, en modifiant le comptage et en le chronométrant très précisément, de battre rigoureusement la seconde (modifiez le 1000 de la ligne 50 à la demande).

5. Un minuteur

Ce n'est pas mal, mais on est loin d'une horloge. D'abord, il faut compter les secondes jusqu'à 60, et pas au-delà. Qu'à cela ne tienne : on va ajouter une instruction disant que *si* on atteint 60, *alors* on remet les secondes à zéro. Sachez que *si* se dit « *if* » en anglais, et que *alors* se dit « *then* ». Voici ce que cela donne :

```
]10 rem...les secondes...
]20 s=0
]30 HOME
]40 ? TAB(16);s
]50 FOR T=0 TO 1000: NEXT
]60 s=s+1
]70 IF s=60 THEN GOTO 20
]80 GOTO 30
```

Si s est arrivé à 60, alors on retourne en 20 le remettre à zéro

Lancez-le : cette fois, le compte atteint 59 que l'on voit nettement affiché, mais dès qu'il passe à 60, il est ramené à zéro.

IF... THEN

Lorsque le Basic rencontre une instruction IF... THEN, il l'examine :

- **Si la condition posée (ici s = 60) est remplie, il exécute ce que lui impose le then (ici : GOTO 20).**

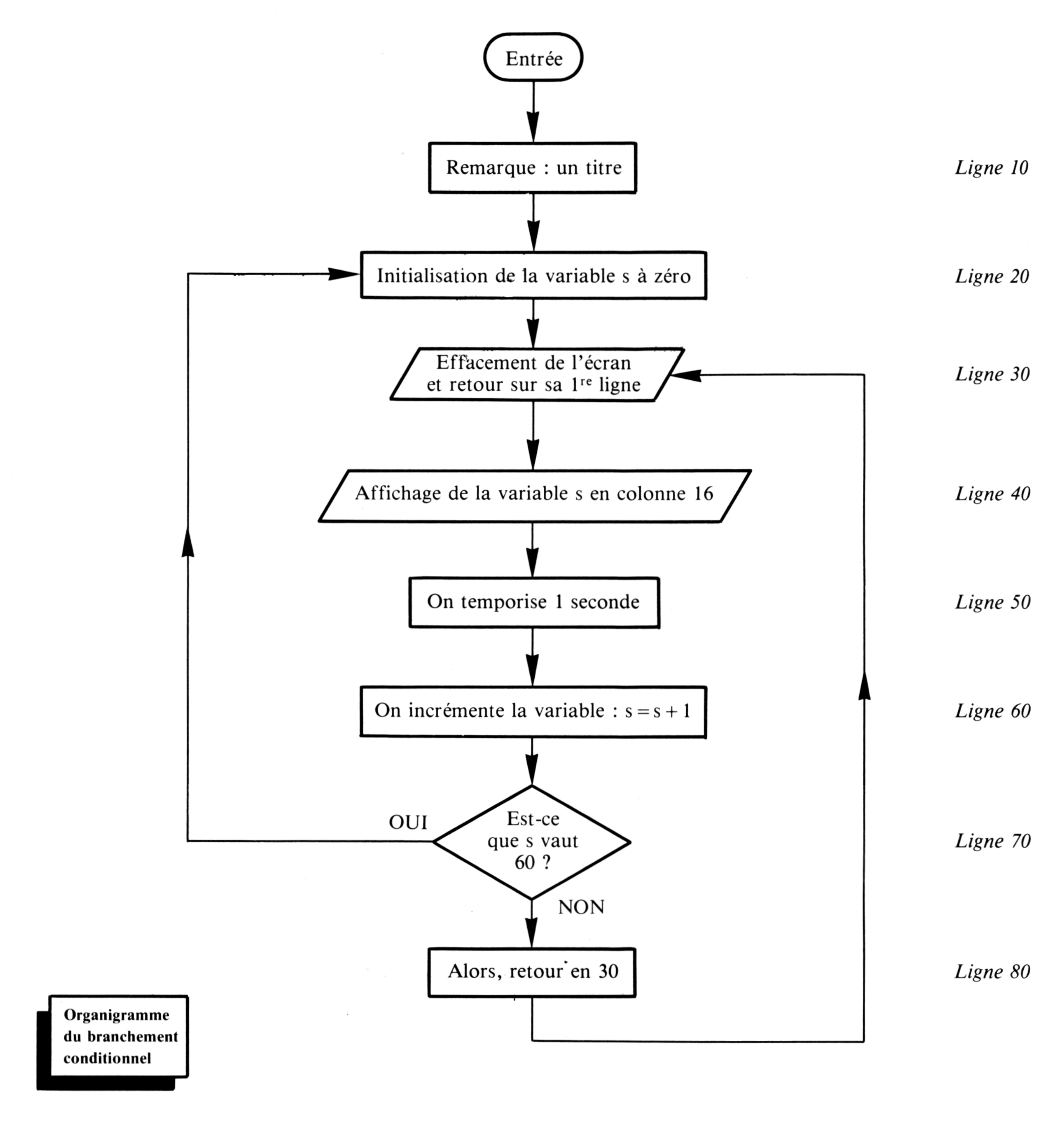
- **Si elle n'est pas remplie, il ignore purement et simplement ce IF... THEN et passe à la ligne suivante (donc, ici, la 70).**

Une condition remplie est dite *vraie* dans le langage informatique ; si elle n'est pas vérifiée, elle est dite *fausse.* Ainsi, si la condition est fausse, le Basic traite la ligne IF... THEN aussi bien qu'une REM : il l'ignore. Le IF... THEN offre toutefois une possibilité nouvelle : celle d'exécuter un branchement *sur condition.*

BRANCHEMENT CONDITIONNEL

On dit que le IF... THEN est une instruction de branchement conditionnel. Cela, par opposition au GOTO qui est un branchement inconditionnel.

On peut le dessiner dans un organigramme (voir page suivante).



Organigramme du branchement conditionnel

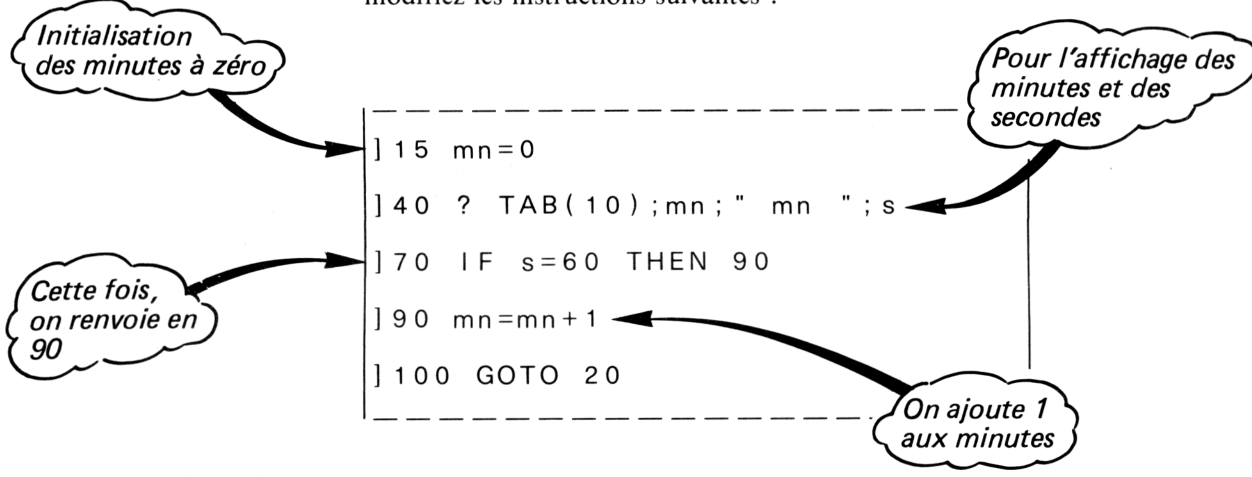
En pratique, un THEN suivi d'un GOTO est parfaitement inutile. On peut supprimer purement et simplement le GOTO et tout fonctionne aussi bien.

```
10 REM ...les secondes...
20 s = 0
30 HOME
40 PRINT TAB(16); s
50 FOR t = 0 TO 1000: NEXT
60 s = s+1
70 IF s = 60 THEN 20
80 GOTO 30
```

D'ailleurs, on pourrait conserver le GOTO et se passer du THEN, dans ce cas, ce qui donnerait :

```
10 REM ...les secondes...
20 s = 0
30 HOME
40 PRINT TAB(16);s
50 FOR t = 0 TO 1000: NEXT
60 s = s+1
70 IF s = 60 GOTO 20
80 GOTO 30
```

Mais pourquoi ne pas compter aussi les minutes ? Ajoutez à votre programme, ou modifiez les instructions suivantes :



MINUTEUR

Le IF a renvoyé, cette fois, en 90 pour que le processus soit plus évident. En 90, on incrémente en effet les minutes et on passe en 100 qui renvoie en 20. Le listage du programme donne :

```
list

   10 REM  minutes et secondes
   15 mn = 0
   20 s = 0
   30 HOME
   40 PRINT TAB(10); mn; " mn "
; s
   50 FOR t = 0 TO 1000: NEXT
   60 s = s+1
   70 IF s = 60 THEN 90
   80 GOTO 30
   90 mn = mn+1
  100 GOTO 20
```

Lancez-le ; vous verrez s'inscrire, en haut de l'écran, les minutes et secondes, par exemple :

```
3 mn 25
```

Nous vous laissons le soin de poursuivre le développement de cette horloge et de régler par vous-même le problème des heures !

6. Quelles conditions ?

Il nous faut toutefois ajouter que, dans un ordre IF... THEN, la condition peut être autre chose que l'égalité ou l'identité. Voici la liste des *relations* possibles avec leurs opérateurs :

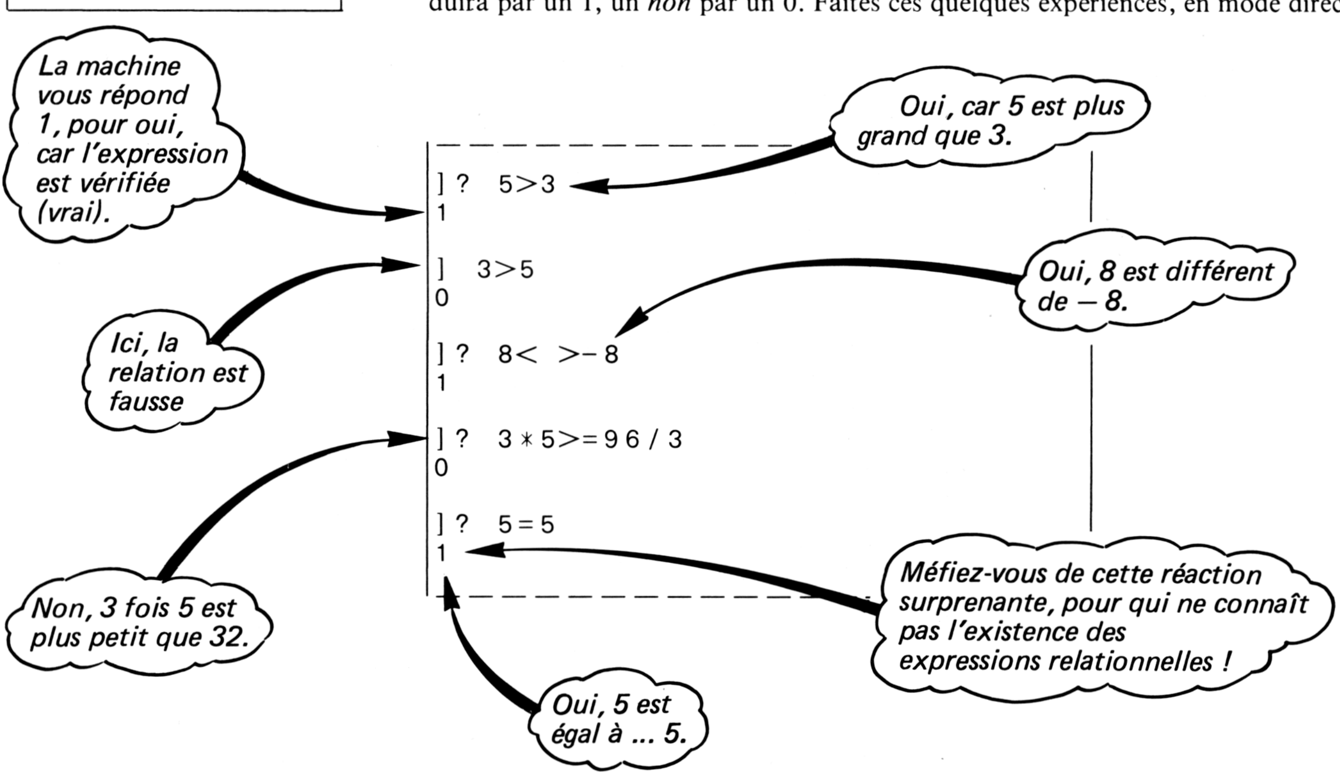
OPERATEURS RELATIONNELS

OPERATEURS RELATIONNELS

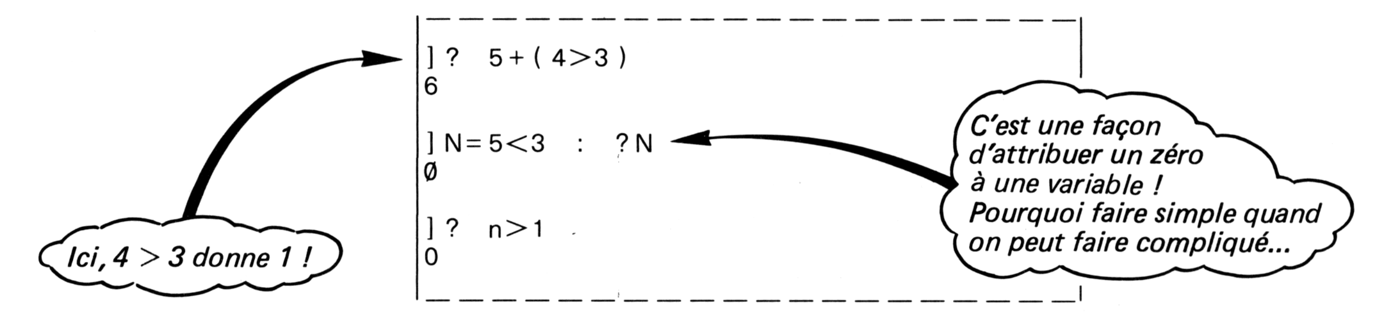
Notation Basic	Signification	Notation classique
=	Egal	=
<	Plus petit que	<
>	Plus grand que	>
>=	Plus grand ou égal	⩾
<=	Plus petit ou égal	⩽
<>	Différent de	≠

Vrai = 1
Faux = 0

On remarquera que ces symboles sont disponibles au clavier, alors que certains des symboles mathématiques équivalents, d'usage courant, exigent des claviers spéciaux. Cependant, il est intéressant de noter que Basic sait répondre à une équation relationnelle, par oui ou par non, selon qu'elle est vérifiée ou non. Un *oui* se traduira par un 1, un *non* par un 0. Faites ces quelques expériences, en mode direct :



Dans ce dernier cas, en effet, le symbole = n'est pas un symbole d'affectation (il n'y a pas d'affectation), mais un *symbole relationnel :* 5 = 5. Or, cette relation est *vraie,* donc l'ordinateur répond « 1 ».
Ainsi qu'on peut en juger, l'emploi de ces expressions est parfois déconcertant ; en voici d'autres exemples :



Plus fort encore ; devinez à quoi peut bien être égal le b défini ci-après :

```
] b=5432=2345
```


Si vous hésitez, demandez la réponse à ADAM :

```
]? b
0
```

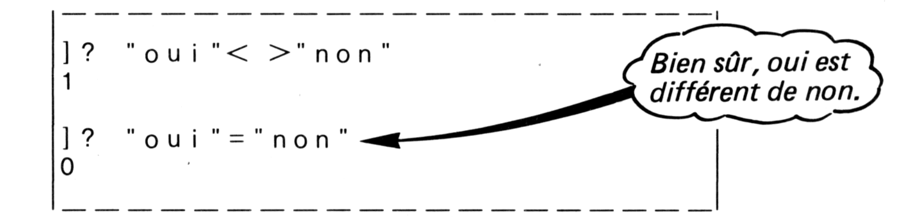
C'est le même cas que ci-dessus, où l'on demandait PRINT 5 = 5 et où la réponse était 1, car l'égalité est vérifiée. Ici, la relation est manifestement fausse, car 5432 n'égale pas 2345 ; donc, la machine répond 0.

RELATIONS DE CHAINES ALPHANUMERIQUES

Ces symboles servent également à la comparaison de chaînes (ce qui sera utile, par exemple, pour dresser une liste de noms par ordre alphabétique ; mais n'anticipons pas). Dans ce cas, ce sera l'identité ou l'ordre alphabétique qui interviendront, avec les significations suivantes :

Opérateur	Signification	Exemple
=	Est identique à	"Bonjour" = "Bonjour"
< >	Précède dans l'ordre alphabétique Suit dans l'ordre alphabétique	"A"<"B" "C">"B"
<=	Précède ou est identique	"A"<="B" "B"<="B"
>=	Suit ou est identique	"C">="B" "C">="C"
<>	Est différent de	"A"<>"B"
+	*(Nous ajoutons, ici, en outre, ce symbole)* Symbole de concaténation	"O" + "U" + "I" *(ce qui donne OUI)*

Faisons quelques essais avant d'aller plus loin :



Les chaînes de caractères doivent *toujours* figurer entre guillemets !

ASCII

Comment s'effectue l'étude de la relation ? Tout d'abord, le Basic se reporte à un tableau général de tous les caractères. Ce tableau a été normalisé sous le nom de ASCII (pronconcez : *aski),* ce qui signifie *"Americain Standard Code For Information Interchange" ;* à chaque caractère possible (et il y en a davantage que sur le clavier) est attribué un *poids* (un nombre). Ce *poids* sert au classement. Reportez-vous à ce tableau, en fin d'ouvrage : vous verrez que les majuscules commencent en 64 et les minuscules en 96 : donc, le *poids* de minuscules est supérieur de 32 à celui des majuscules.

Majuscules et minuscules, lorsqu'elles sont incluses dans des chaînes, sont prises en compte avec leur poids propre. On ne devra plus utiliser n'importe quoi...

Vous remarquerez aussi que le code ASCII comporte de nombreux autres types de caractères.

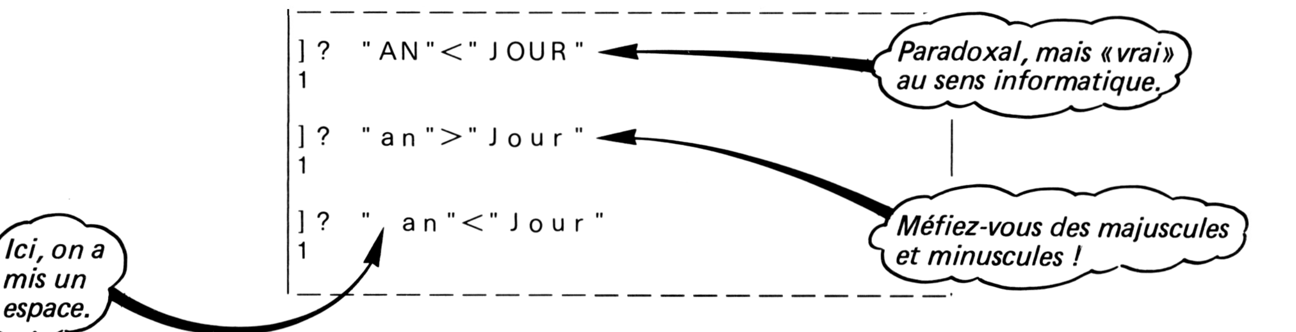
COMPARAISON DE CHAINES

Constatez, également, que le poids des chiffres est moindre que celui des lettres puiqu'ils commencent en 48. Par conséquent, un classement alphanumérique accordera la préséance aux chiffres.

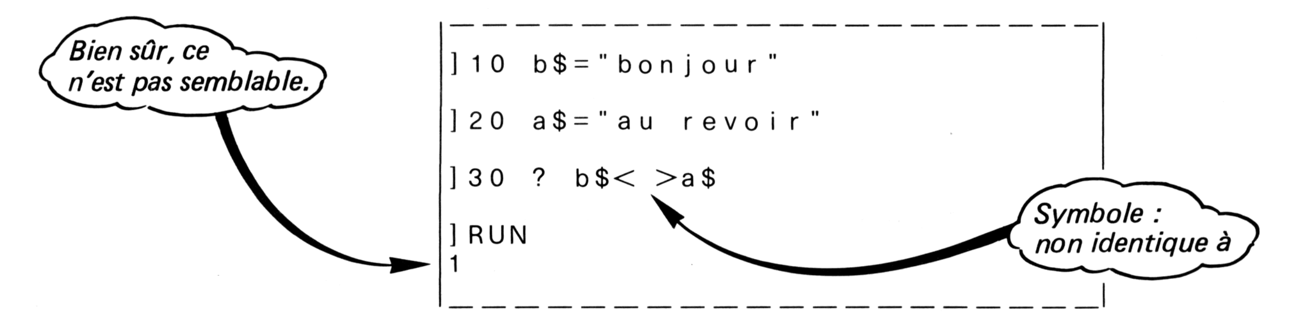
Pour comparer deux chaînes, le Basic comparera le premier caractère de chacune d'elles en comparant le poids respectif de ces caractères, puis éventuellement les seconds, puis les troisièmes, etc. S'ils sont tous identiques, les chaînes seront réputées *égales.* La première comparaison qui détectera une différence servira à classer les chaînes. Si, au cours des comparaisons, et alors qu'on a toujours obtenu des identités, une chaîne se termine avant l'autre, la plus courte sera classée avant la plus longue.

Attention encore : ici, les espaces inclus dans la chaîne comptent ! Leur poids est 32.

Voici l'exemple de quelques relations entre chaînes *vraies* (l'ordinateur accordera généreusement un 1 si on lui pose la question).



Ainsi, un classement alphabétique donnera la prépondérance aux majuscules sur les minuscules. On pourrait utiliser des noms de variables chaînes dans ce même but :



Dans l'exemple ci-dessus, remarquez bien la différence entre les noms des variables que nous avons choisis (suffixés par $ pour des variables chaînes) et les noms des constantes chaînes, mises entre guillemets. Sinon, comment s'y retrouver...

NOT
AND
OR

Il nous reste encore à examiner le cas des opérateurs logiques. On aborde, ici, des *fonctions logiques* telles que les ET (en anglais, AND), le NON, ou inversion (NOT) et le OU (OR, en anglais). Ces fonctions sont étudiées dans les cours élémentaires de logique... ; en voici les conclusions, toujours avec 1 pour « vrai » et 0 pour « faux », et pour deux variables logiques A et B.

OPERATEURS LOGIQUES

Supposons que les deux variables logiques soient : A	B	**La fonction AND donnera**	**La fonction OR donnera**
Ø	Ø	Ø	Ø
1	Ø	Ø	1
Ø	1	Ø	1
1	1	1	1

(1 = vrai, et Ø = faux)

L'inversion est plus simple, car :

Si une variable est :	**L'inversion (NOT) donnera**
Ø	1
1	Ø

Nous n'insisterons guère sur ces opérateurs, car il faudrait développer ici leurs règles de fonctionnement et par conséquent, réaliser un cours de logique. Donnons cependant quelques exemples d'application :

```
]? 5=5 and 3=3
1

]? 5=5
1

]? NOT 5=5
0

]? NOT 5=3
0

]? 5=5 OR 3=5
1
```

L'une ET l'autre des égalités sont vraies.

not, ou NOT, peu importe.

L'une OU l'autre des égalités est vraie.

Méfiez-vous de ce genre d'égalité banale qui donne 1 en résultat... le non-initié est toujours surpris !

C'est l'inversion : aussi, le 1 devient un zéro

A vous de continuer ce genre d'exercice.

7. Enseignement assisté par ordinateur

Voici, pour conclure ce chapitre, un programme de ce que l'on appelle *l'enseignement assisté par ordinateur,* ou EAO en abrégé. Il servira à apprendre la table de multiplication et... à faire aussi le point dans la mesure où il reprend nombre d'instructions examinées jusqu'ici.

ENSEIGNEMENT ASSISTE PAR ORDINATEUR
EAO

Pour obtenir le listage de ce programme sur imprimante (il fait plus d'un écran), nous avons frappé :

```
]pr#1
```

C'est ce qui explique que cette fois, parce que nous n'avons pas recopié l'écran, nous avons travaillé sur des lignes de 60 caractères. Pour annuler l'action de PR#1, il faut faire PR#0.

Dans ce programme, *m* est le multiplicande, *n* le multiplicateur tiré au hasard entre 0 et 9, et *p* le produit proposé. Notez aussi qu'à la question de la ligne 110, il faut répondre par un *o* pour *oui* ou *n* pour *non* (la réponse est affectée à la variable *r*$). Voici donc ce listage :

```
 10 REM  * multiplications *
 20 HOME
 25 INPUT "Tu veux apprendre  la table des ... ? "; m
 30 HOME
 35 n = INT(RND(9)*10)
 40 PRINT "Combien font "; m; " fois "; n
 50 INPUT p
 55 PRINT: PRINT
 60 IF p = m*n GOTO 100
 80 PRINT "Mais non, ce n'est pas cela ! La bonne réponse est "; m*n
 90 GOTO 105
100 PRINT "Bravo, mais c'est très bien pour un débutant !"
105 PRINT: PRINT
110 PRINT "Un autre essai (o/n) ?"
120 INPUT r$
130 IF r$ = "o" THEN 30
135 PRINT: PRINT
140 PRINT "Bien, merci et au revoir !"
```

Voici ce que pourrait donner son lancement ; on a choisi la table de 7 avec deux réponses. Après avoir reçu les félicitations de la machine, on sort du jeu avec un *n,* mais vous auriez pu frapper n'importe quoi d'autre qu'un *o* car on ne vérifie pas cette dernière frappe :

```
Tu veux apprendre la table des
... ? 7

Combien font 7 fois 7
?41


Mais non, ce n'est pas cela ! L
a bonne réponse est 49


Un autre essai (o/n) ?
?o


Combien font 7 fois 4
?28


Bravo, mais c'est très bien pou
r un débutant !


Un autre essai (o/n) ?
?n


Bien, merci et au revoir !
_______________________________
```

Essayez de bien comprendre la logique de ce programme, que nous avons voulu davantage pédagogique que bien fignolé. Remarquez aussi que nous avons deux fois commandé des sauts de lignes doubles. Puis, modifiez-le à votre gré pour l'améliorer.

Réponses aux questions sur le chapitre VII

1. Faux : il faut faire intervenir un PRINT et écrire :
PRINT PDL(0)

2. Vrai : l'affichage retourne le code ASCII du caractère frappé sur le clavier numérique et non ce caractère. Or, le 5 a pour code ASCII le nombre 53.

3. Pas d'erreur : le sous-programme peut fort bien être situé *avant* le programme principal. On ne vous l'avait pas dit mais cela va de soi...

4. Horreur : le RETURN ne doit *jamais* être suivi d'un numéro de ligne. Il est assez grand garçon pour savoir de lui-même où il doit aller.

5. Deux erreurs d'un coup :
• le HTAB est un ordre distinct du PRINT,
• la tabulation doit venir entre parenthèses.
On aurait été mieux inspiré d'écrire :

```
10 HTAB(12)
20 PRINT « etc. »
```

6. Erreur : SPC, lui, doit venir dans un PRINT. Il fallait mettre :
PRINT SPC(12);« etc. »

7. Oui, c'est normal et il n'y a pas d'erreur. Il peut fort bien y avoir une ligne zéro ; et s'il n'y en a pas, c'est probablement parce que t ne prendra jamais les valeurs 1, ou 2, ou 3, et qu'on n'utilisera que les adresses 4 (donc 345) ou 5 (donc 678).

Réponses aux questions sur le chapitre IX

1. Mais c'est tout à fait correct : les DATA peuvent se trouver après un END.

2. Là aussi, il n'y a pas d'erreur, qu'alliez-vous donc penser : RESTORE restaure la liste à son début.

3. La haute définition porte sur 160 par 256 points.

4. En haute définition, il faut employer HPLOT.

5. Oui, on peut tracer des droites avec HPLOT x,y TO...

6. Oui, en supprimant les 4 lignes de texte du bas, donc en utilisant HGR2, ce qui donne 192 lignes sur 256 colonnes.

7. Oui, on peut écrire directement dans une cellule avec POKE.

8. On agrandit un dessin en HGR avec SCALE.

9. La rotation est obtenue avec ROT.

Questions sur le chapitre VI

1. Pour obtenir une valeur aléatoire, vous écrirez :
   ☐ PRINT RND
   ☐ ? RND(1985)

2. L'ordre INT(3.14) vous donnera :
   ☐ 3
   ☐ 4

3. Avec PRINT INT(RND(9) *10) + 1 vous obtiendrez des valeurs entre :
   ☐ 1 et 10
   ☐ 0 et 11

4. Avec la séquence suivante :
   10 IF 5>3 THEN 30
   20 ? "BONJOUR" : END
   30 ? "Au revoir"
   vous afficherez :
   ☐ BONJOUR
   ☐ Au revoir

5. L'ordre suivant T = 14 = 15 provoquera l'affichage de :
   ☐ 0
   ☐ Un message d'erreur

6. Que donnera l'ordre : ? NOT 5 = 7
   ☐ 0
   ☐ 1

Réponses page 132

CHAPITRE VII

SOUS-PROGRAMMES

On va commencer par examiner comment se servir des claviers numériques avant d'aborder un autre des concepts fondamentaux de la programmation, celui des sous-programmes. Tout cela va être entouré d'un ensemble de notions complémentaires.

Principaux thèmes étudiés

Claviers numériques	Appel d'une touche
Manettes de jeux	GET
PDL	HTAB
Levier directionnel	VTAB
Poussoirs des blocs numériques	SPC
Défilé des 16 couleurs	Branchement calculé
Sous-programme	ON... GOTO
Programme principal	Branchement calculé à sous-programme
GOSUB	ON... GOSUB
RETURN	

CLAVIERS NUMERIQUES

MANETTES DE JEUX

PDL

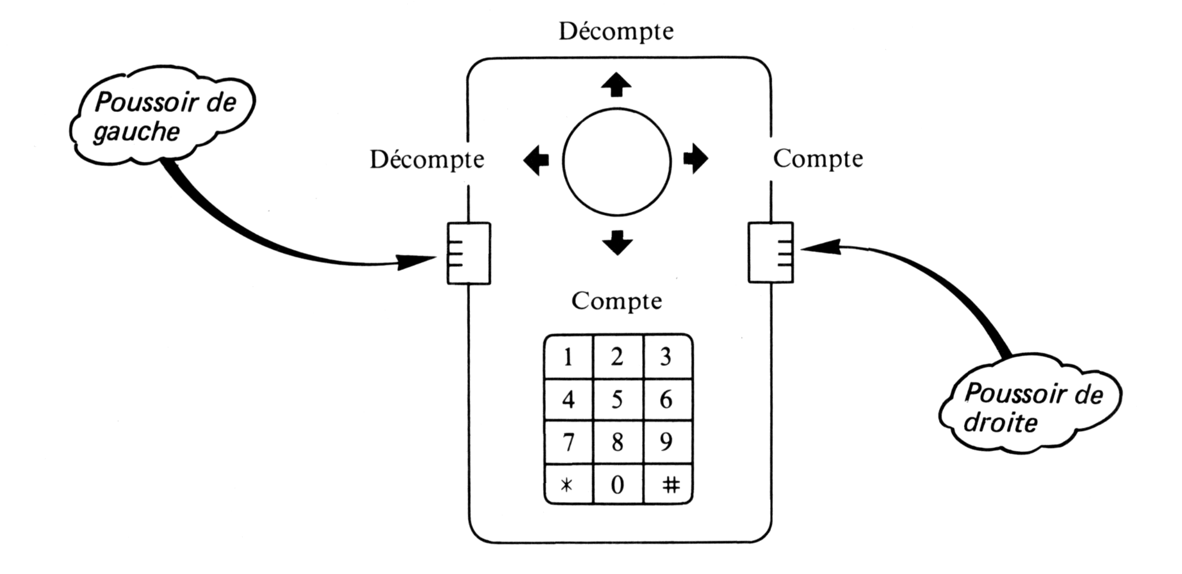
1. Les claviers numériques

A votre système sont associés deux claviers numériques distincts, comportant un « levier directionnel » et deux poussoirs. Les Américains utilisent le mot « paddle » pour désigner les manettes, servant non seulement aux jeux mais à toutes sortes d'autres choses ; de là vient l'ordre PDL que nous allons utiliser. La liste de ses possibilités se trouve à la page suivante.

Ils se font donc sur deux axes, verticalement et horizontalement. Le comptage va de 0 à 255 selon la *durée* d'un maintien en position, et dans le sens indiqué par nos flèches.
Rien de plus facile que de vérifier cela. Entrez le programme suivant. La ligne 3 crée deux sauts de ligne ; on va afficher sur deux colonnes les états du levier directionnel numéro 0 : l'état vertical pour la colonne de gauche, l'état horizontal pour celle de droite (un retour au repos du levier directionnel, au centre donc, conserve les états acquis). La ligne 8 trace un tireté, pour la beauté de la présentation. Enfin, la ligne 10 affiche la valeur acquise par les leviers directionnels, 20 renvoyant en 10.

PDL	Affiche l'état	Commentaires
PDL (0)	Du levier 0, verticalement	Les deux manettes/blocs numériques sont référencés 0 et 1
PDL (1)	Du levier 1, verticalement	
PDL (2)	Du levier 0, horizontalement	
PDL (3)	Du levier 1, horizontalement	
PDL (4)	Direction, levier 0	Haut = 1, bas = 4 droite = 2, gauche = 8
PDL (5)	Direction, levier 1	
PDL (6)	Du poussoir de gauche, clavier 0	0 pour pousoir au repos, 1 pour poussoir pressé
PDL (7)	Du poussoir de gauche, clavier 1	
PDL (8)	Du poussoir de droite, clavier 0	
PDL (9)	Du poussoir de droite, clavier 1	
PDL (10)	La valeur de la touche pressée, clavier 0	En code ASCII (48 pour le 0, 49 pour le 1, 50 pour le 2, etc., 57 pour 9, 42 pour * et 35 pour #)
PDL (11)	La valeur de la touche pressée, clavier 1	
PDL (12)	Clavier 0, touche # et *	* = 10, # = 11 Aucune = 15
PDL (13)	Clavier 1, touche # et *	
PDL (14)	Clavier 0	Non encore affecté
PDL (15)	Clavier 1	

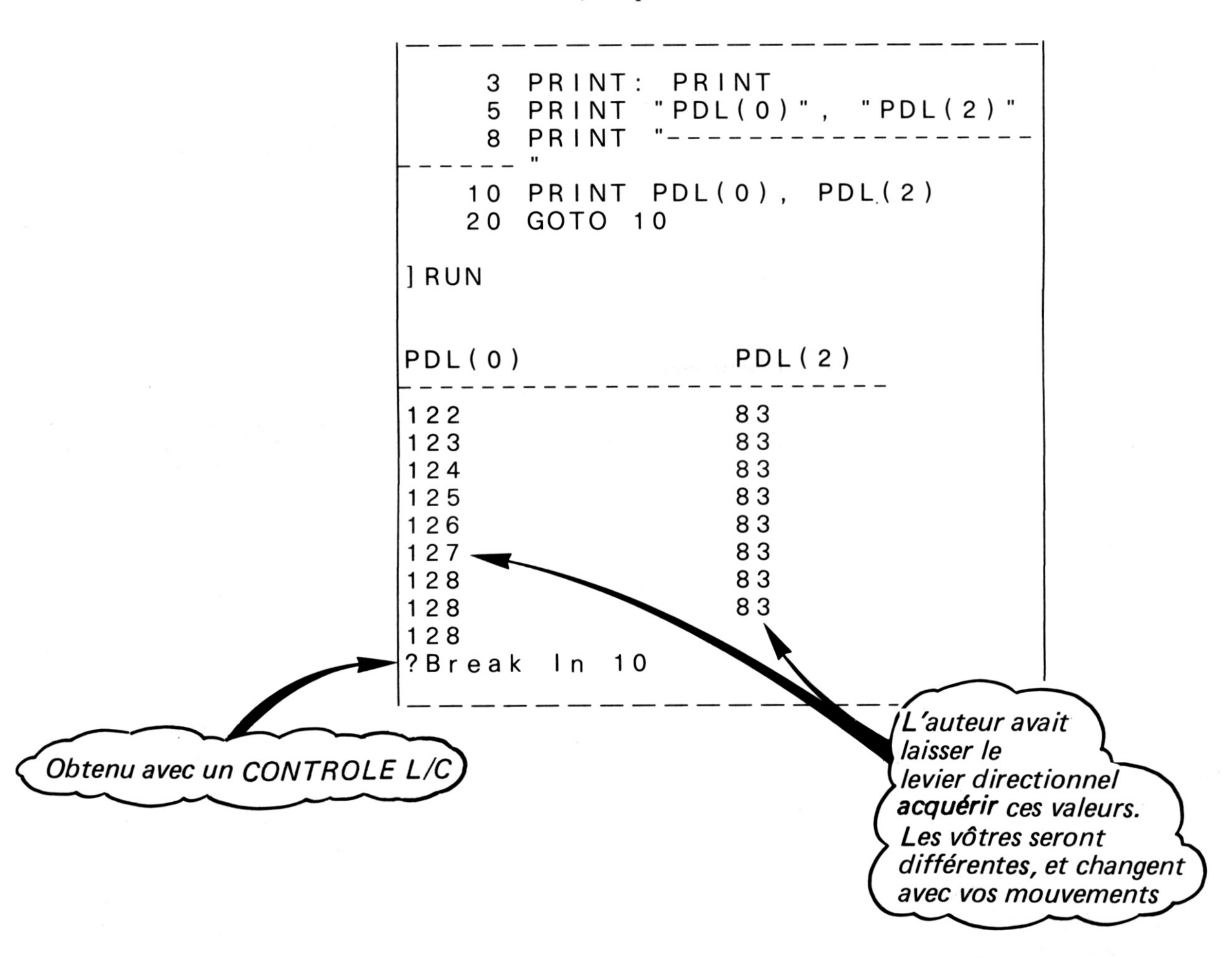
Les mouvements de levier directionnel (des manettes de jeux) sont les suivants :



LEVIER DIRECTIONNEL

Une poussée et un maintien du levier directionnel 0 en position *haute* provoquera un décomptage jusqu'à 0, et en position *basse,* un comptage jusqu'à 255, affiché sur la colonne de gauche. Horizontalement, le décomptage se fait à *gauche* et le comptage à *droite,* toujours de 0 à 255, pour PDL(2), sur la colonne de droite, donc.

Avec ce programme, les chiffres vont défiler ; il faudra donc le stopper avec un CONTROLE/C, ce que nous avons fait avec cet essai :



```
   3 PRINT: PRINT
   5 PRINT "PDL(0)", "PDL(2)"
   8 PRINT "------------------------
"
  10 PRINT PDL(0), PDL(2)
  20 GOTO 10

]RUN

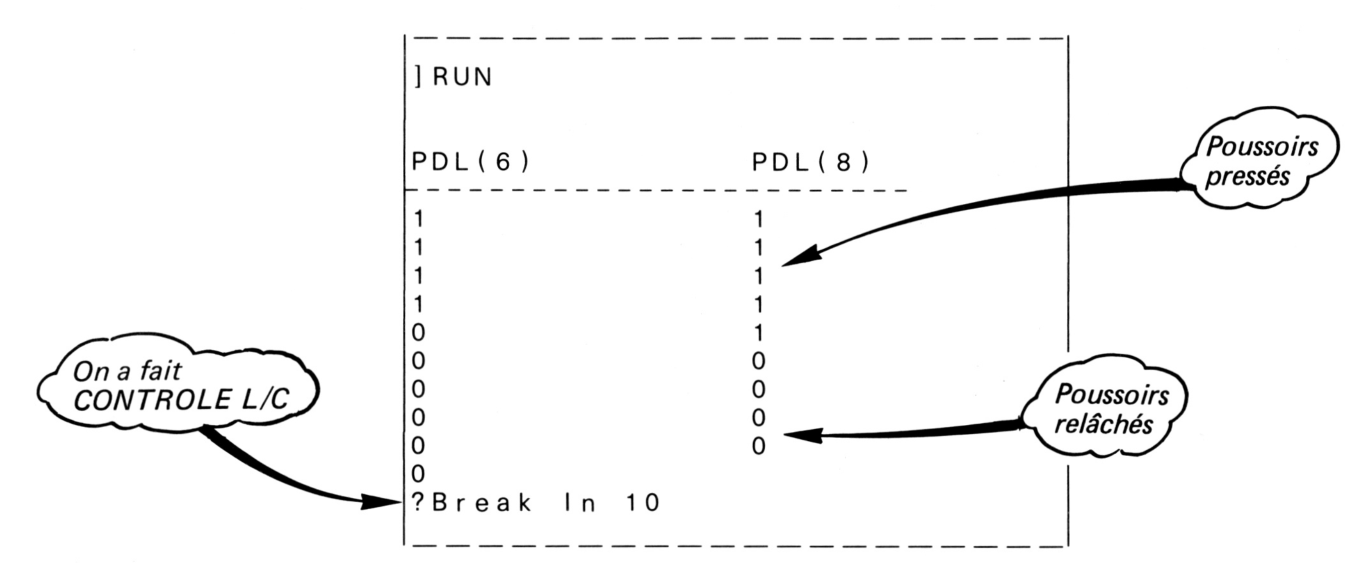
PDL(0)             PDL(2)
------------------------------
122                83
123                83
124                83
125                83
126                83
127                83
128                83
128                83
128
?Break In 10
```


Relancez ce programme et manœuvrez les leviers directionnels vers le haut et vers le bas, en les maintenant quelques instants en position. Vous allez voir les comptages ou décomptages s'exécuter.

POUSSOIRS DES BLOCS NUMERIQUES

Faites-en autant avec la direction, ou les positions des poussoirs. Vous obtiendrez quelque chose comme cela (l'auteur a maintenu les poussoirs fermés puis les a relâchés) :

```
   3 PRINT: PRINT
   5 PRINT "PDL(6)", "PDL(8)"
   8 PRINT "------------------------
"
  10 PRINT PDL(6), PDL(8)
  20 GOTO 10
```

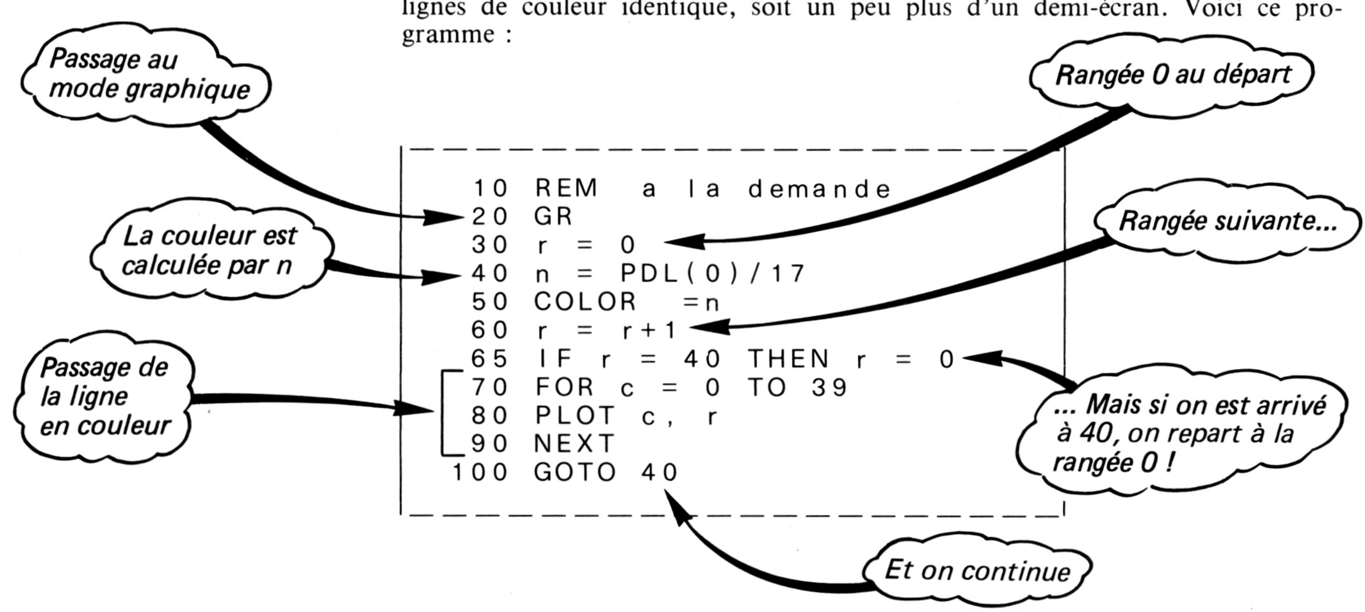


Essayez maintenant avec les valeurs correspondant aux claviers numériques, afin de bien vous convaincre que cela fonctionne parfaitement, avec des PDL(10) ou (11), vous recevrez le numéro de code ASCII du caractère frappé sur ce clavier *et non ce caractère ;* pour revenir au caractère, vous pourriez demander PRINT CHR$ (code), ainsi qu'on l'a déjà vu.

2. Défilé des couleurs

DEFILE DES COULEURS

Pour vérifier le fonctionnement de PDL, nous allons écrire un programme qui va permettre de faire défiler les 16 couleurs, en avant ou en arrière, selon la position de PDL(0). Celui-ci fournissant toutes les valeurs de 0 à 255, on va diviser ce nombre par 17 pour obtenir de 0 à 15 qu'on va affecter à COLOR (rappelez-vous que les 16 couleurs sont numérotées de 0 à 15 et qu'on les obtient en faisant COLOR = l'un de ces numéros). La division indiquée va fournir des valeurs décimales mais le Basic, avec son infinie bonté, n'en retiendra que la partie entière (ce qu'il fait d'ailleurs souvent). Dans ce cas, plusieurs divisions successives vont donner le même résultat, par exemple de 34/17 à 50/17 qui, toutes, donneront 2 (l'entier du résultat) ; cela se reproduira 50 – 34 = 26 fois, ce qui explique qu'on aura jusqu'à 26 lignes de couleur identique, soit un peu plus d'un demi-écran. Voici ce programme :



Lancez ce programme, puis manœuvrez le levier directionnel en le maintenant en position : vous allez faire défiler les couleurs sur l'écran et, enfin, pouvoir les différencier ! Sachez que d'un téléviseur à l'autre, ces couleurs peuvent différer.

Remarquez aussi la ligne 65 où l'on a écrit que, *si r* était arrivé à 40 et puisqu'il ne peut aller que jusqu'à 39 (les rangées sont numérotées de 0 à 39), il fallait repartir au début et lui ré-attribuer zéro. C'est la double égalité de cette instruction qui lui confère un aspect curieux, mais c'est parfaitement légal. La preuve : ça marche.

3. Les sous-programmes

SOUS-PROGRAMME

Un *sous-programme,* c'est tout simplement un programme distinct auquel on peut avoir recours chaque fois que besoin s'en fait sentir. Dans ce cas, on « l'appelle » par le numéro de sa première ligne d'instruction. Pour le différencier, le programme qui appelle un sous-programme est désigné par le nom de *programme principal.*

PROGRAMME PRINCIPAL

Pour en examiner le fonctionnement, on va recourir à un exemple très simple avec, cette fois, le poussoir gauche du bloc numérique 0 ; on le « lit » avec PDL(6) ; il est à 0 au repos, mais passe à 1 lorsqu'il est pressé. Notre programme va faire défiler des zéros mais affichera « Feu ! » en tabulant à 12 lorsque vous presserez ce poussoir. Sa seule originalité consiste à faire appel à un sous-programme.

Notons deux choses :

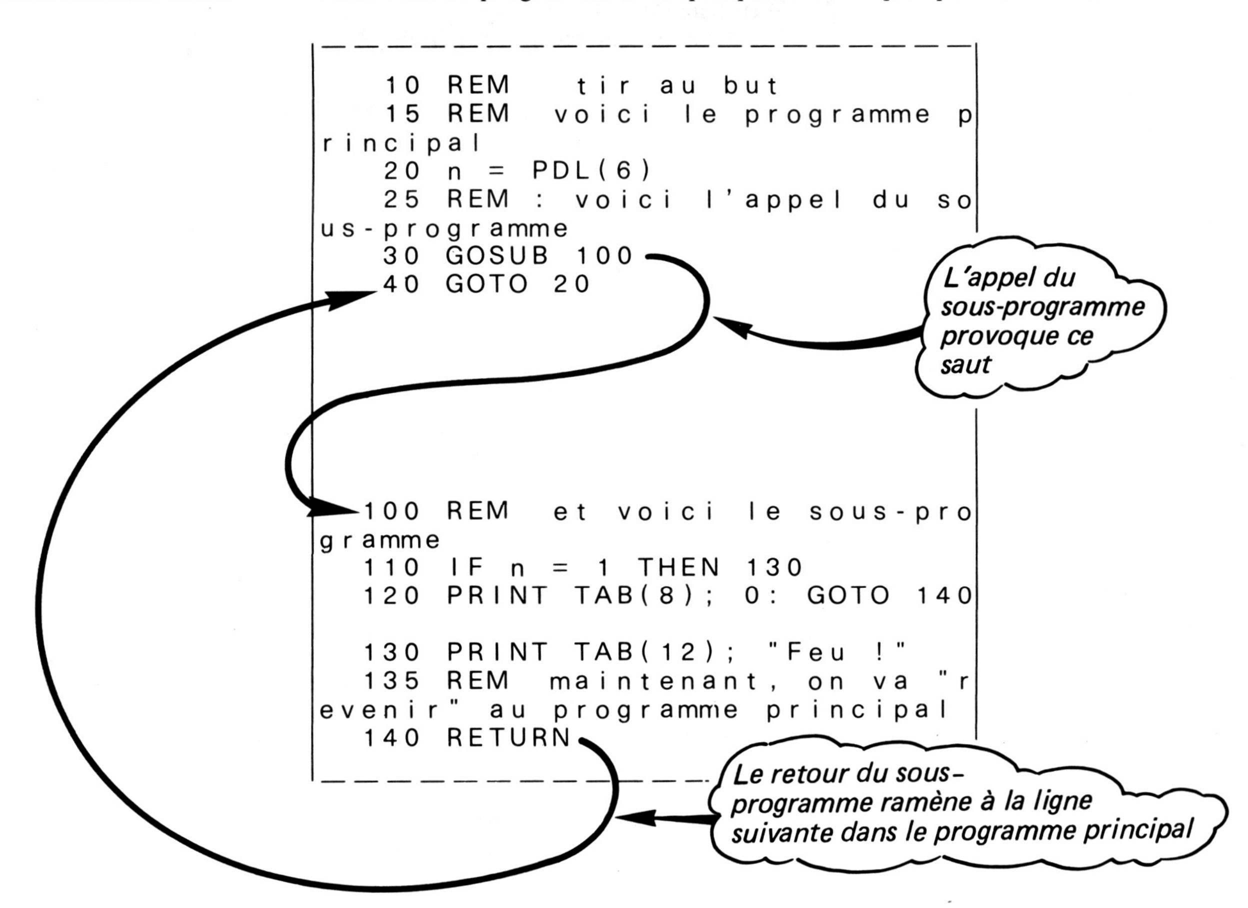
GOSUB

1) L'appel du sous-programme se fait via l'ordre GOSUB, de l'anglais « go to subroutine », soit « aller au sous-programme » ;

RETURN

2) Le retour du sous-programme au programme principal, celui qui l'a appelé, se commande RETURN, mot anglais signifiant « retour ».

Voici donc ce programme avec quelques REMarques por vous aider :

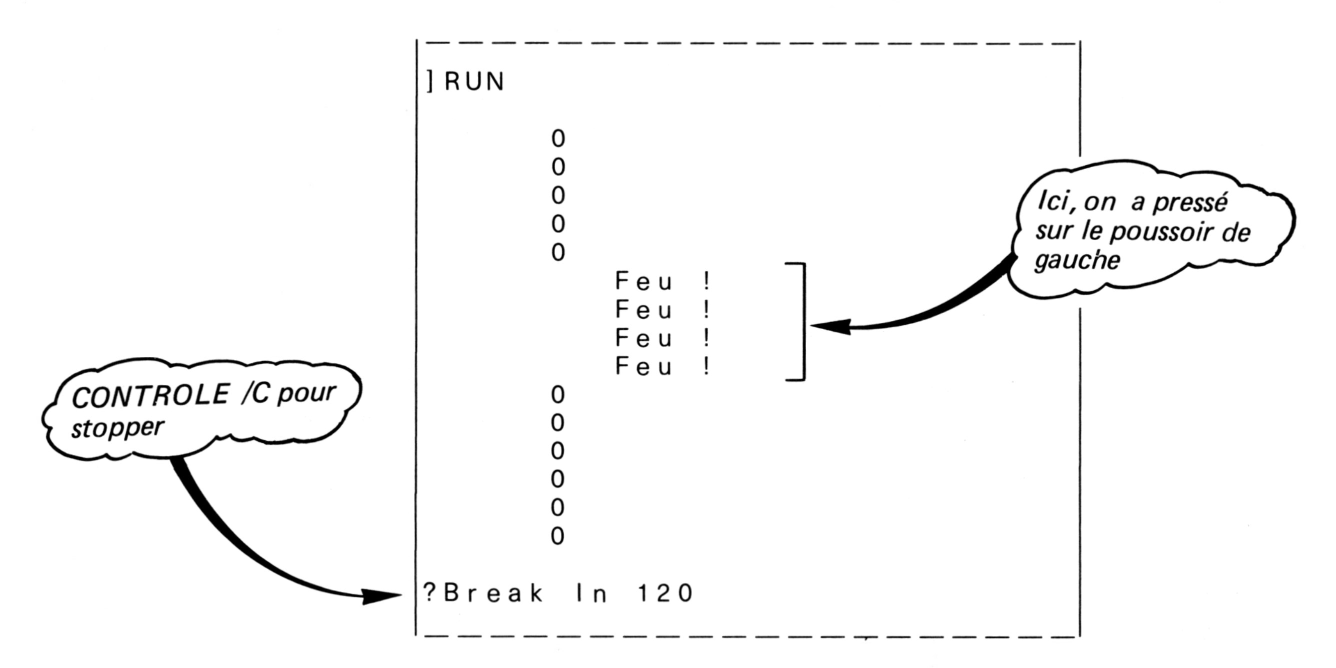


```
   10 REM   tir au but
   15 REM  voici le programme p
rincipal
   20 n = PDL(6)
   25 REM : voici l'appel du so
us-programme
   30 GOSUB 100
   40 GOTO 20

  100 REM  et voici le sous-pro
gramme
  110 IF n = 1 THEN 130
  120 PRINT TAB(8); 0: GOTO 140

  130 PRINT TAB(12); "Feu !"
  135 REM  maintenant, on va "r
evenir" au programme principal
  140 RETURN
```

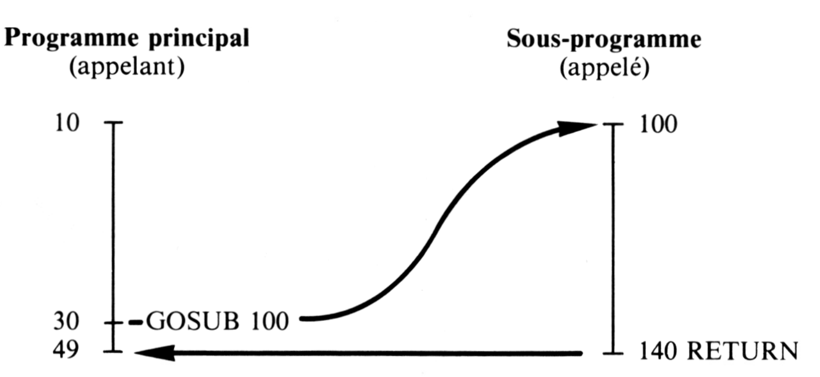

Avant d'aller plus loin, voici ce que provoquera un RUN :



Voyons maintenant comment tout cela fonctionne, avec deux règles de base :

> **• L'appel du sous-programme GOSUB doit être suivi par le numéro de l'instruction qui commence le sous-programme (il pourrait y avoir plusieurs sous-programmes... ; c'est donc le moyen de ne pas les mélanger).**
> **• Le RETURN n'a pas besoin de numéro d'instruction : il se fait obligatoirement à l'instruction qui suit l'appel GOSUB.**

On peut illustrer graphiquement ce processus de la façon suivante :



La longueur des programmes ne joue ici aucun rôle. Sachez qu'on peut « appeler » autant de fois qu'on le désire un sous-programme, de n'importe quel point du programme principal : le RETURN gèrera toujours automatiquement le retour à l'instruction qui suit celle d'appel.
D'autre part, un programme principal peut disposer de plusieurs sous-programmes. Chacun de ceux-ci peut, à son tour, avoir une progéniture et appeler son, ou ses sous-sous-programmes, bref, on peut faire aussi compliqué qu'on le veut ! Bien que ce soit souvent un moyen de faire simple.

> **De toutes façons, le retour se fait *obligatoirement* au programme appelant.**

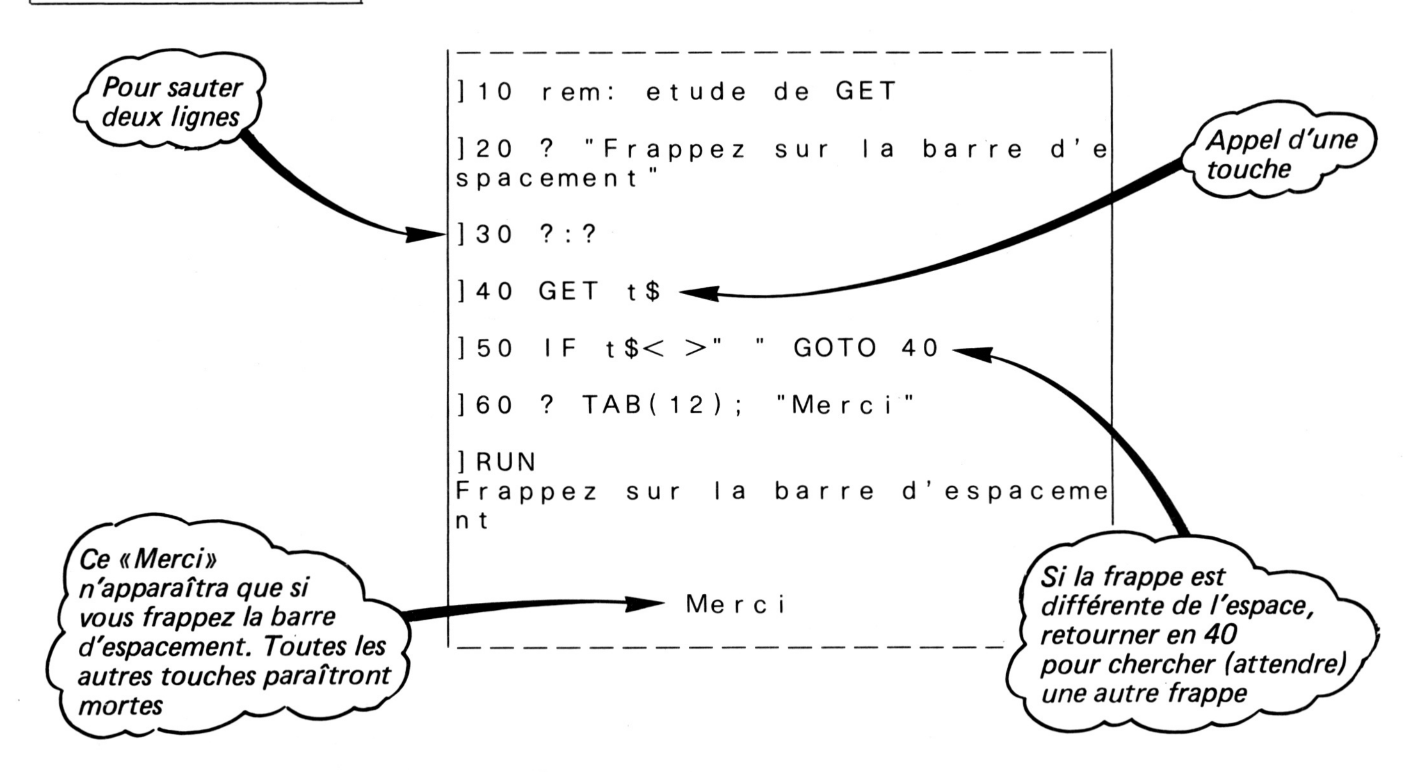
Il faut donc qu'à chaque GOSUB corresponde un RETURN.

APPEL D'UNE TOUCHE

GET

4. L'appel d'une touche

Parce que nous allons nous servir de cet ordre par la suite, il convient de l'étudier ici ; c'est l'ordre GET, qui sert à « appeler une touche » quelconque qu'on désignera globalement par T$ (on la considère comme une chaîne de caractères). Lorsque le Basic rencontre un GET, il stoppe son action et attend qu'une touche soit frappée ; dès lors, il daigne poursuivre l'exécution du programme. Cela donnera, avec un essai simple :



```
]10 rem: etude de GET

]20 ? "Frappez sur la barre d'e
spacement"

]30 ?:?

]40 GET t$

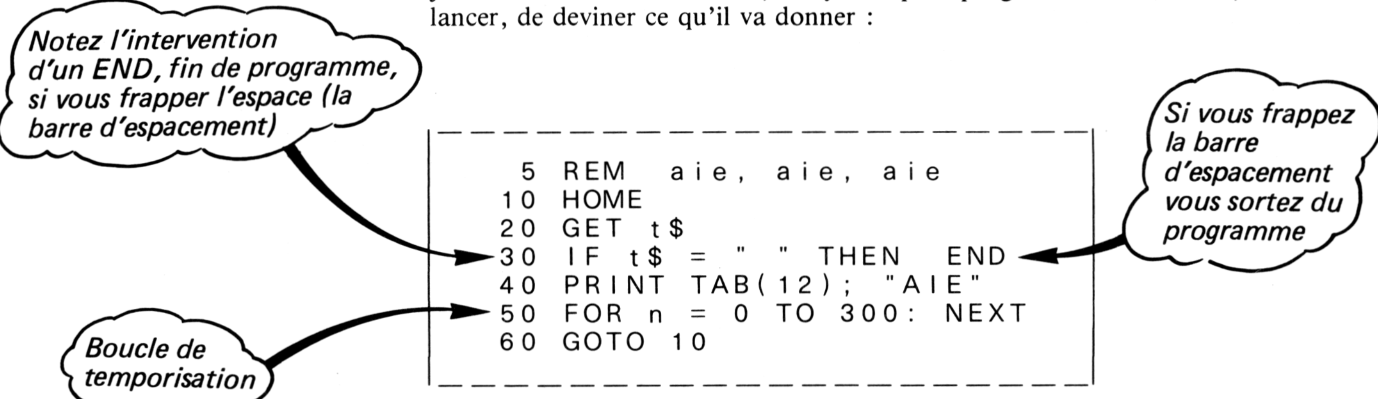
]50 IF t$< >" " GOTO 40

]60 ? TAB(12); "Merci"

]RUN
Frappez sur la barre d'espaceme
nt


            Merci
```


Tant que le programme tourne, avant que vous ne frappiez sur la barre d'espacement, toutes les autres touches du clavier semblent mortes, y compris CONTROLE/C ; méfiez-vous donc de ce genre de programme et prévoyez toujours une sortie. En variante, essayez ce petit programme mais tentez, avant de le lancer, de deviner ce qu'il va donner :



```
 5 REM aie, aie, aie
10 HOME
20 GET t$
30 IF t$ = " " THEN END
40 PRINT TAB(12); "AIE"
50 FOR n = 0 TO 300: NEXT
60 GOTO 10
```


Oui, à chaque fois que vous frapperez sur une touche quelconque, un AIE s'inscrira, en haut et au centre de l'écran, pour un court instant.

Notez bien que GET attend une, et une seule touche, et il n'a pas besoin de la frappe de RETURN pour l'acquérir. C'est ce qui le différencie de l'ordre INPUT.

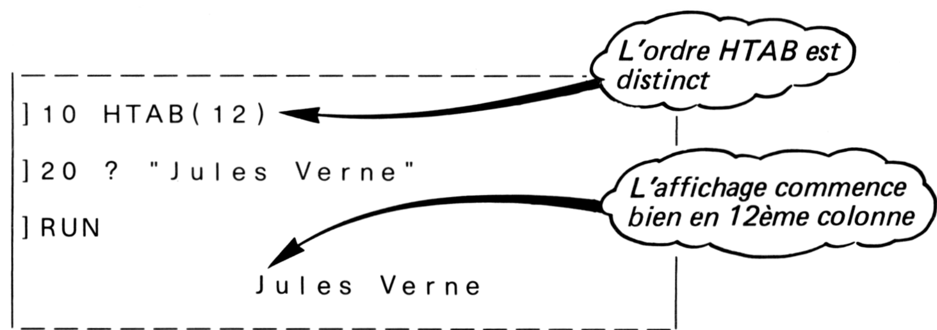
5. Pour positionner le curseur

On a utilisé des PRINT TAB dans les programmes précédents, mais vous devez savoir que, pour afficher sur un emplacement précis de l'écran, tant horizontal que vertical, on dispose encore de diverses instructions.
Commençons par HTAB, pour *tabulation horizontale.* On fait suivre HTAB du numéro de la colonne où l'on veut que le curseur se précipite ; ce sera là où commencera l'affichage. Ce numéro doit être fourni entre parenthèses.

HTAB

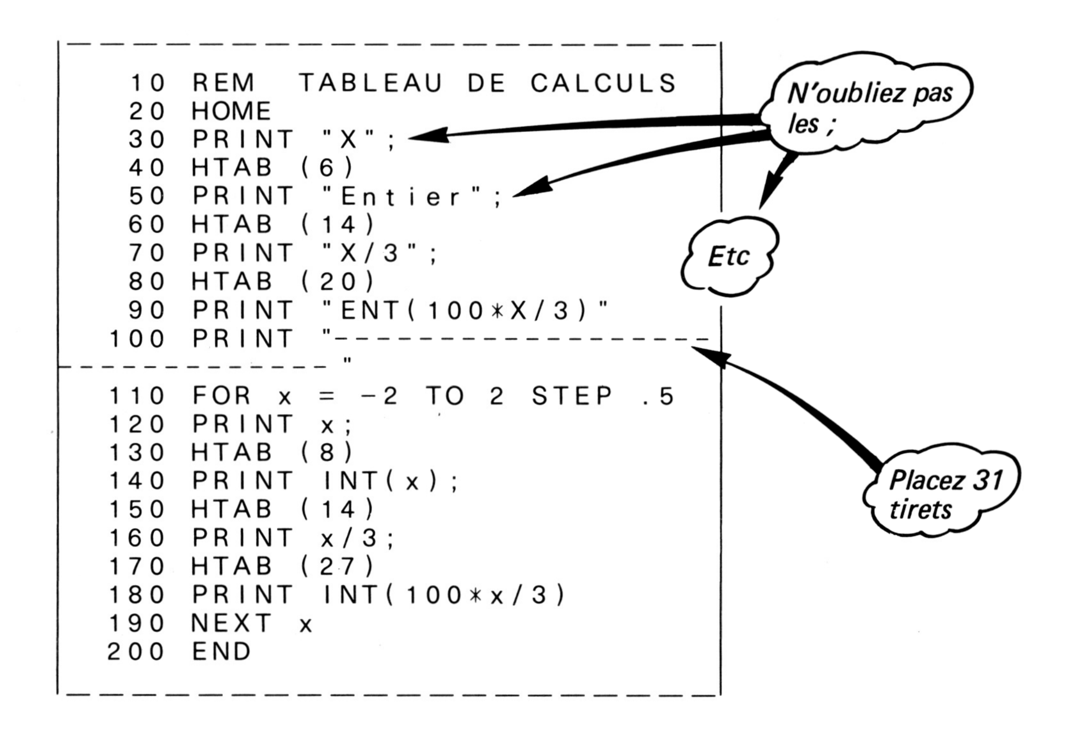
> **Attention, HTAB, tout comme VTAB, plus loin, doivent être considérés comme des ordres distincts du PRINT.**

Par exemple, on écrira ce court programme qui écrit *Jules Verne* à partir de la douzième colonne :



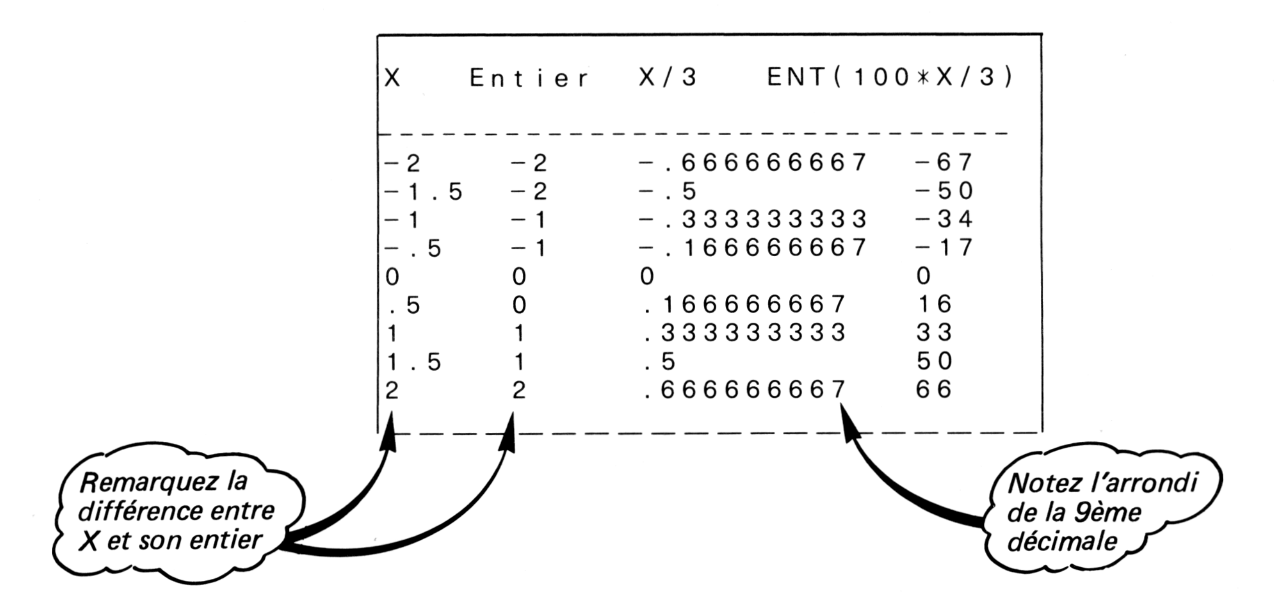
```
]10 HTAB(12)
]20 ? "Jules Verne"
]RUN
           Jules Verne
```


Utilisons cette tabulation pour mettre en place une démonstration de l'extraction d'entiers de diverses valeurs. On va créer un tableau, qui n'est pas encore la perfection, mais vous verrez qu'on est sur la bonne voie. Les valeurs de HTAB choisies sont celles qui procurent un résultat acceptable. Toutes les instructions restent séparées (si vous vous trompez, vous les corrigerez plus aisément !). On va donc afficher X, de −2 à 2 et par pas de 0.5, puis son entier, le tiers de X, et enfin l'entier de cent fois cette dernière valeur :



```
 10 REM  TABLEAU DE CALCULS
 20 HOME
 30 PRINT "X";
 40 HTAB (6)
 50 PRINT "Entier";
 60 HTAB (14)
 70 PRINT "X/3";
 80 HTAB (20)
 90 PRINT "ENT(100*X/3)"
100 PRINT "-----------------
-----------"
110 FOR x = -2 TO 2 STEP .5
120 PRINT x;
130 HTAB (8)
140 PRINT INT(x);
150 HTAB (14)
160 PRINT x/3;
170 HTAB (27)
180 PRINT INT(100*x/3)
190 NEXT x
200 END
```

Sur ordre RUN, l'écran s'efface en raison du HOME et de la ligne 20 et vous obtenez :



X	Entier	X/3	ENT(100*X/3)
-2	-2	-.666666667	-67
-1.5	-2	-.5	-50
-1	-1	-.333333333	-34
-.5	-1	-.166666667	-17
0	0	0	0
.5	0	.166666667	16
1	1	.333333333	33
1.5	1	.5	50
2	2	.666666667	66

Ce HTAB peut fournir des résultats intéressants, en outre, si on apprend à bien le maîtriser. Par exemple :

```
    10 FOR i = 0 TO 6
    20 HTAB (3)
    30 PRINT "****"
    40 NEXT

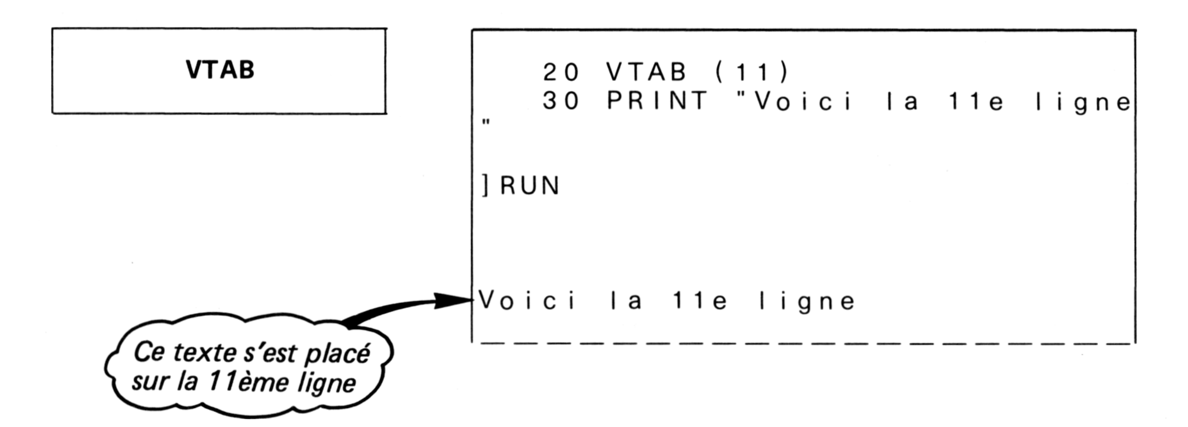
]RUN
  ****
  ****
  ****
  ****
  ****
  ****
  ****
```

Ou encore, en modifiant la seule ligne 20 :

```
    10 FOR i = 0 TO 6
    20 HTAB (3*i+6)
    30 PRINT "****"
    40 NEXT

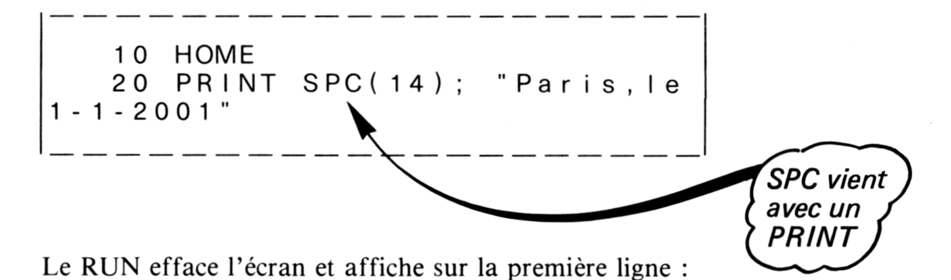
]RUN
     ****
        ****
           ****
              ****
                 ****
                    ****
                       ****
```

En symétrie avec HTAB intervient VTAB, pour *tabulation verticale*. Voici un ordre d'affichage sur la onzième ligne :



Une autre façon de tabuler horizontalement consiste à créer des espaces avec l'ordre SPC, suivi du nombre d'espaces entre parenthèses *et sur une ligne PRINT,* cette fois :

SPC



Le RUN efface l'écran et affiche sur la première ligne :

```
                 Paris,le 1-1-2001
```

Un début de lettre pourrait être, en mélangeant tout cela :

```
   10 HOME
   20 PRINT SPC(14);"Paris,le
1-1-2001"
   30 VTAB (3)
   40 PRINT SPC(5); "Monsieur"
   50 VTAB (5)
   60 PRINT " Nous vous...etc."
```

Ce qui donne, au RUN :

```
                 Paris,le 1-1-2001

     Monsieur

 Nous vous...etc.

]
```

Notez que SPC se place *après* le dernier caractère affiché et non d'une façon absolue, à partir du bord. Vérifiez-le :

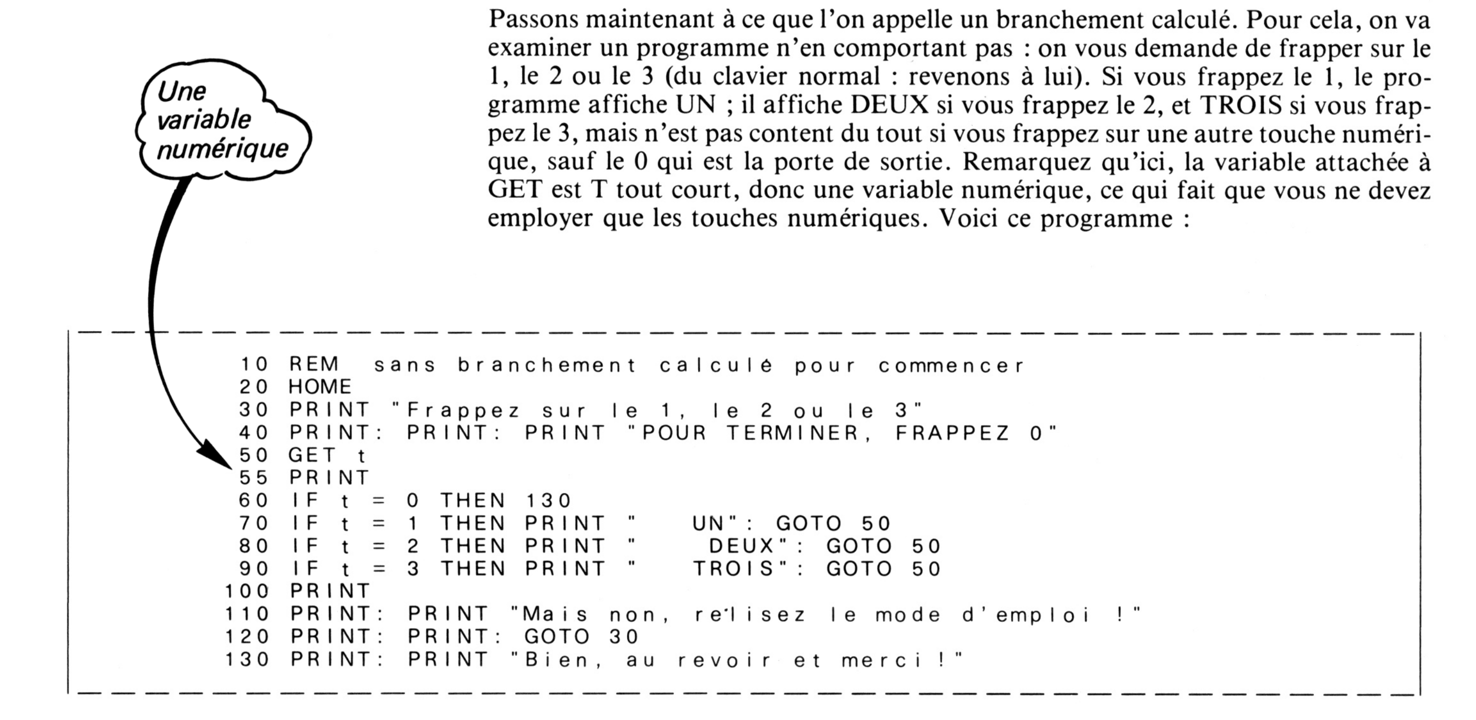
```
]? "Jules"; SPC(5); "Verne"
Jules     Verne

]
```

L'espace entre Jules et Verne est de 5

6. Branchements et appels calculés

Passons maintenant à ce que l'on appelle un branchement calculé. Pour cela, on va examiner un programme n'en comportant pas : on vous demande de frapper sur le 1, le 2 ou le 3 (du clavier normal : revenons à lui). Si vous frappez le 1, le programme affiche UN ; il affiche DEUX si vous frappez le 2, et TROIS si vous frappez le 3, mais n'est pas content du tout si vous frappez sur une autre touche numérique, sauf le 0 qui est la porte de sortie. Remarquez qu'ici, la variable attachée à GET est T tout court, donc une variable numérique, ce qui fait que vous ne devez employer que les touches numériques. Voici ce programme :




```
 10 REM  sans branchement calculé pour commencer
 20 HOME
 30 PRINT "Frappez sur le 1, le 2 ou le 3"
 40 PRINT: PRINT: PRINT "POUR TERMINER, FRAPPEZ 0"
 50 GET t
 55 PRINT
 60 IF t = 0 THEN 130
 70 IF t = 1 THEN PRINT "     UN": GOTO 50
 80 IF t = 2 THEN PRINT "      DEUX": GOTO 50
 90 IF t = 3 THEN PRINT "     TROIS": GOTO 50
100 PRINT
110 PRINT: PRINT "Mais non, re'lisez le mode d'emploi !"
120 PRINT: PRINT: GOTO 30
130 PRINT: PRINT "Bien, au revoir et merci !"
```

Voici ce que donnerait un lancement, où l'on aurait frappé 3, puis 2, puis 1, puis 3, et enfin 0 :

```
Frappez sur le 1, le 2 ou le 3

POUR TERMINER, FRAPPEZ 0

    TROIS

     DEUX

    UN

    TROIS

Bien, au revoir et merci !
```

Vous avez frappé sur le zéro

Si vous frappez sur une autre touche que les chiffres, et parce que la variable est numérique, vous obtiendrez (on a frappé ici sur une lettre alphabétique) :

```
Frappez sur le 1, le 2 ou le 3

POUR TERMINER, FRAPPEZ 0
?Syntax Error In 50
```

« Erreur de syntaxe à la ligne 50 »

En effet, c'est bien à la ligne 50 que figure la variable numérique. Ainsi, le Basic achève là-dessus ce programme, le fait « avorter », comme l'on dit en informatique.
Cela vu, transposons ce programme à l'usage d'un branchement calculé. C'est simple : il suffit de demander au Basic d'examiner ce qu'on a frappé pour la variable *t* ; si c'est 1, on lui commande d'aller à la première adresse indiquée sur la même ligne ; si c'est 2, à la seconde, et à la troisième si c'est 3. La syntaxe de cet ordre est :

BRANCHEMENT CALCULE

ON... GOTO

ON *t* GOTO *première adresse,deuxième,troisième...*

On peut placer autant d'adresses que l'on veut, mais restons-en à 3. Voici ce que cela donne avec le même programme ; on a supprimé les lignes 80 et 90, et remplacé la ligne 70 par une nouvelle qui fournit le branchement calculé :

```
 10 REM  avec un branchement calculé, cette fois
 20 HOME
 30 PRINT "Frappez sur le 1, le 2 ou le 3"
 40 PRINT: PRINT: PRINT "POUR TERMINER, FRAPPEZ 0"
 50 GET t
 55 PRINT
 60 IF t = 0 THEN 130
 70 ON t GOTO 121, 122, 123
110 PRINT: PRINT "Mais non, relisez le mode d'emploi !"
120 PRINT: PRINT: GOTO 30
121 PRINT "  UN": GOTO 50
122 PRINT "    DEUX": GOTO 50
123 PRINT "    TROIS": GOTO 50
130 PRINT: PRINT "Bien, au revoir et merci !"
```

Voici le branchement calculé

Ce branchement calculé de la ligne 70 fonctionne ainsi :

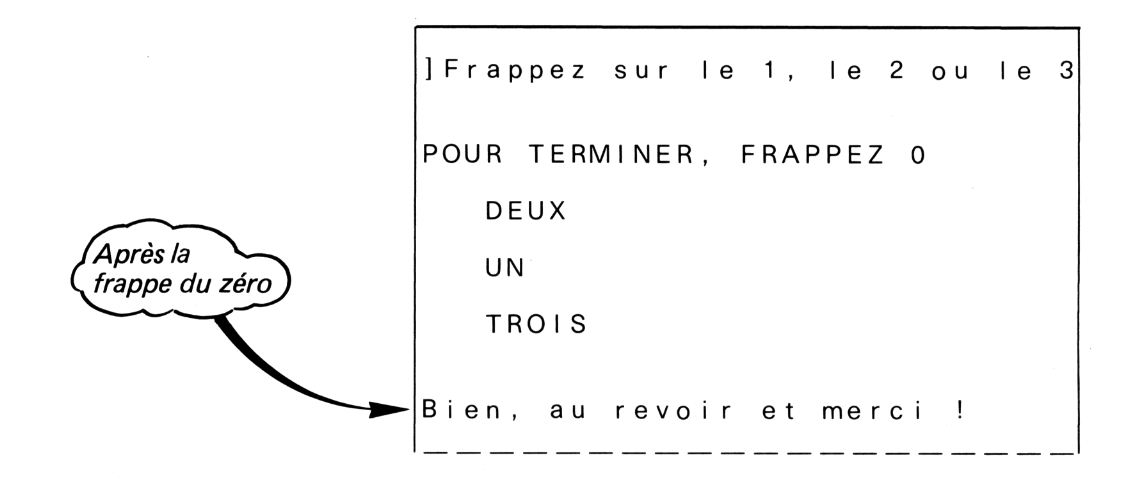
• si *t* vaut 1, le branchement se fait en 121 ;

• si *t* vaut 2, il se fait *à la seconde adresse,* soit 122 ;

• si *t* vaut 3, il se fait *à la troisième adresse,* soit 123 ;

• pour toute autre valeur, les branchements sont ignorés et le programme se poursuit à la ligne suivante, donc la 110.

Notez qu'ici, le 0 organise la sortie ; s'il n'était pas ainsi employé, il provoquerait aussi le passage en 110. Un RUN donnerait à nouveau :




```
]Frappez sur le 1, le 2 ou le 3

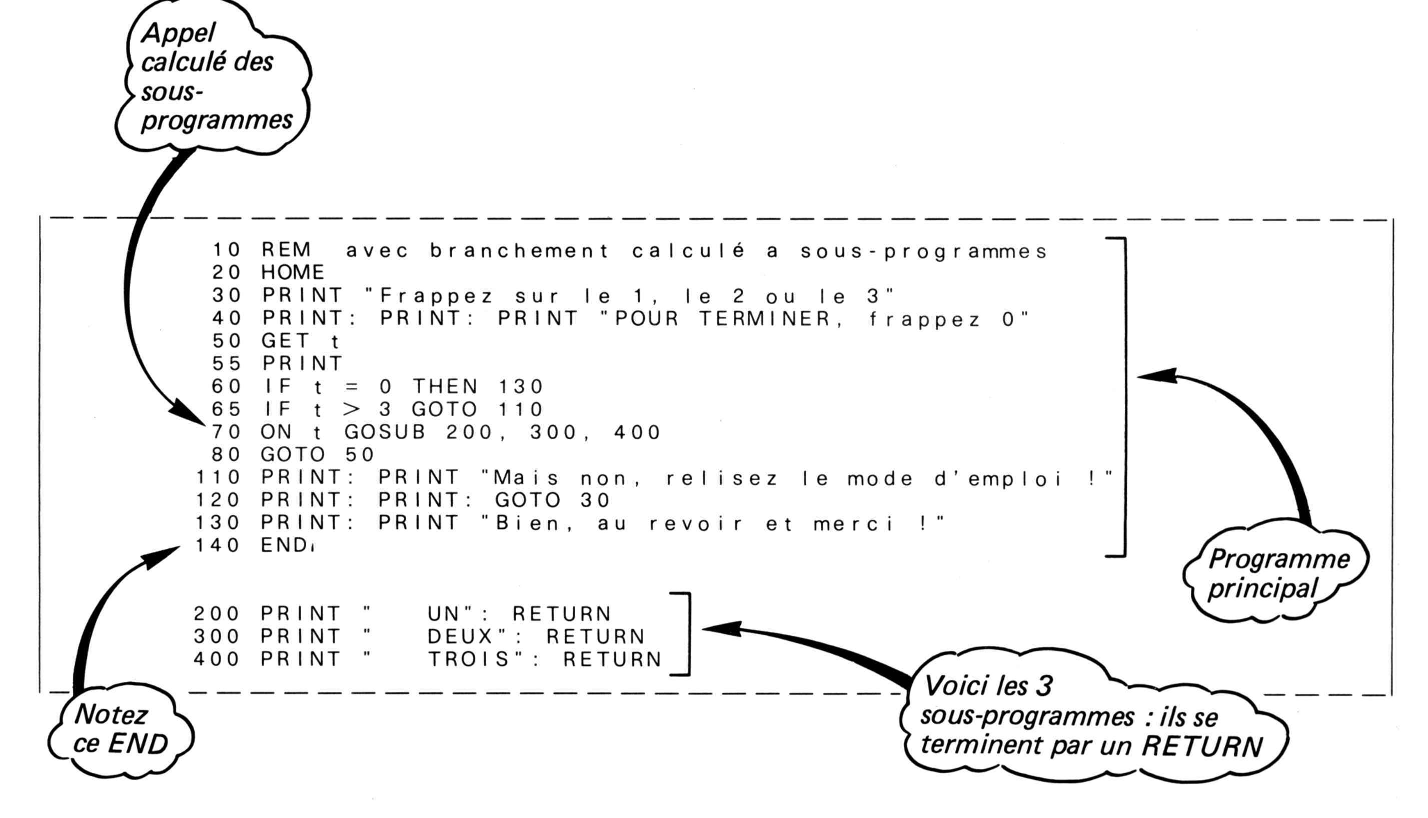
POUR TERMINER, FRAPPEZ 0
    DEUX
    UN
    TROIS

Bien, au revoir et merci !
```

BRANCHEMENT CALCULE A SOUS-PROGRAMME

ON... GOSUB

Le même principe s'applique aux sous-programmes. En conservant le même programme, le branchement commandera le passage à l'un des trois sous-programmes (d'une ligne chacun !) commençant en 200, 300 et 400. On a ajouté, en 65, un test probable : si la touche frappée est supérieure à 3, on passe en 110 afficher le message de protestation. Cela donne :




```
 10 REM  avec branchement calculé a sous-programmes
 20 HOME
 30 PRINT "Frappez sur le 1, le 2 ou le 3"
 40 PRINT: PRINT: PRINT "POUR TERMINER, frappez 0"
 50 GET t
 55 PRINT
 60 IF t = 0 THEN 130
 65 IF t > 3 GOTO 110
 70 ON t GOSUB 200, 300, 400
 80 GOTO 50
110 PRINT: PRINT "Mais non, relisez le mode d'emploi !"
120 PRINT: PRINT: GOTO 30
130 PRINT: PRINT "Bien, au revoir et merci !"
140 END

200 PRINT "   UN": RETURN
300 PRINT "   DEUX": RETURN
400 PRINT "   TROIS": RETURN
```

Le lancement fournira le même résultat que précédemment. Notez que les RETURN renvoient à l'instruction qui suit celle d'appel et par conséquent, en 80, qui renvoie à son tour en 50.

Il a fallu, ici, placer un END à la fin du programme principal ; c'est une sorte de frontière. Sans elle, en effet, et après avoir écrit « Bien, merci et au revoir », le Basic irait tout naturellement à l'instruction suivante qui est, ici, la 200, afficherait « UN » puis tomberait sur un RETURN alors qu'il n'est pas passé par un appel GOSUB : il se fâcherait alors, affichant un message d'erreur, ce qui donnerait dans notre cas (on a supprimé la ligne 140) :

```
Frappez sur le 1, le 2 ou le 3

POUR TERMINER, frappez 0

Bien, au revoir et merci !
   UN

?RETURN without GOSUB Error In
200
```

Le message du Basic : «Je suis tombé sur un RETURN sans GOSUB ; il y a une erreur en ligne 200».

Ce genre de message doit vous inciter à la prudence, car le Basic n'a pas toujours raison à 100 %.

En effet, il ne peut pas détecter qu'il manque une ligne, avec un END, et annonce que c'est la ligne 200 qui comporte une erreur. Or, elle est parfaitement correcte.

Moralité : n'hésitez pas à chercher ailleurs que sur la ligne indiquée une erreur qui existe bel et bien.

Questions sur le chapitre VII

Cette fois, c'est le jeu des erreurs. Trouverez-vous celles qui figurent peut-être dans ces lignes ?

1. Pour lire la commande du levier directionnel PDL(0), on doit écrire :
   10 PDL(0)
2. Si l'on veut lire la touche numérique frappée sur le clavier numérique et si l'on frappe sur le 5, l'affichage donnera :
   53
3. On a écrit la ligne
   1000 GOSUB 30
4. On a écrit :
   RETURN 200
5. On a écrit :
   PRINT HTAB "12"
6. A nouveau, on pose :
   10 SPC(12)
   20 PRINT "Etc"
7. Cela vous semble-t-il normal ?
   ON t GOTO 0,0,0,0,345,678

Réponses page 109

CHAPITRE VIII

L'USAGE DES CASSETTES

Dans ce chapitre, un seul thème va retenir notre attention : l'usage des cassettes digitales pour sauvegarder des programmes.

Principaux thèmes étudiés

CATALOG	LOCK
Fichier	UNLOCK
Sauvegarde	RECOVER
SAVE	MON
Chargement	RENAME
LOAD	NOMON
DELETE fichier	INIT
Verrouillage fichier	

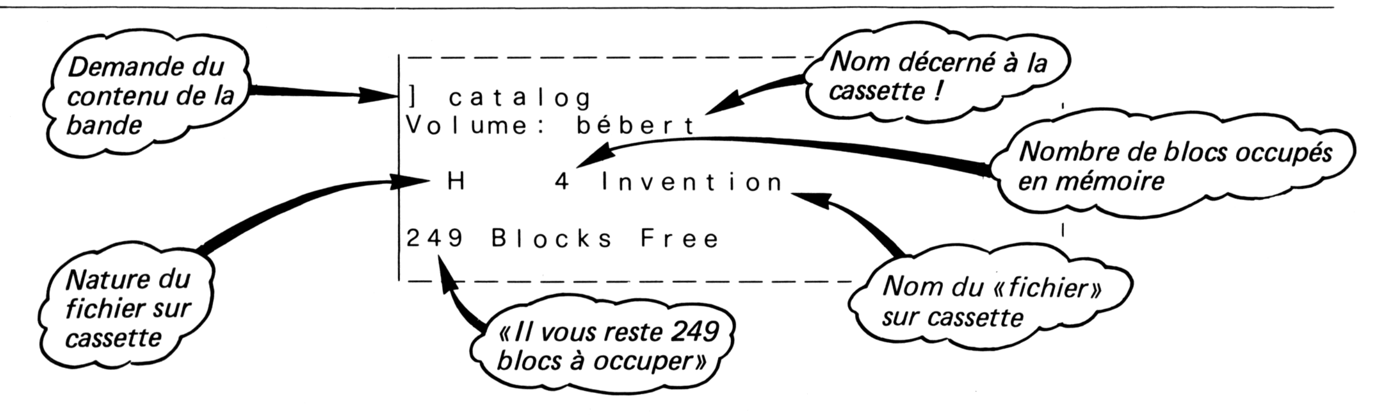
1. Sauvegardons nos programmes

On a vu comment l'on peut sortir un programme sur l'imprimante, mais nous ne vous avons pas encore expliqué comment le ranger sur une cassette. En effet, rappelez-vous que la mémoire interne d'Adam est « volatile » : si l'ordinateur est mis hors tension ne serait-ce qu'un instant, il perd ses informations stockées en mémoire *vive*. Or, vous voudrez peut-être ré-utiliser ultérieurement un programme ; ou encore, vous voudrez le conserver mais libérer la mémoire pour un autre.
Placez la cassette digitale qui vous est livrée vierge avec Adam dans l'unité à cassettes.

Ne vous trompez pas : la face écrite regardant vers vous. Sinon, l'ordinateur ne pourrait pas la lire et vous chercheriez en vain quelle erreur vous avez pu commettre.

CATALOG

Commençons par examiner ce qu'elle contient en demandant, à partir du Basic cette fois et non plus du traitement de texte, sa *table des matières* qui s'appelle ici son *catalogue*. Ecrit à l'anglaise, cet ordre donne (en mode direct, suivi par RETURN) :



Remarquez que vous disposez, sur une cassette, de 253 blocs disponibles, le reste étant occupé par le catalogue. Ici, on a retrouvé le texte « Invention » que nous avions créé avec le traitement de texte. Avant d'aller plus loin, rappelons une définition importante :

FICHIER

On appelle « fichier » une collection d'informations qui se veulent homogènes. Ce pourra être un texte créé par le traitement de texte, un programme, etc.

On va, en effet, utiliser largement ce nom très général de *fichier* car on peut y loger n'importe quoi, ou presque.
Imaginons un autre programme, très simple ; frappez :

```
]10 PRINT "Bonjour"
```

On va le ranger sur cassette ; pour cela, il faut lui décerner un nom, de dix caractères au maximum ; choisissons *Salut*.

Le rangement sur cassette s'appelle la « sauvegarde ». En anglais, ce mot se dit « save ».

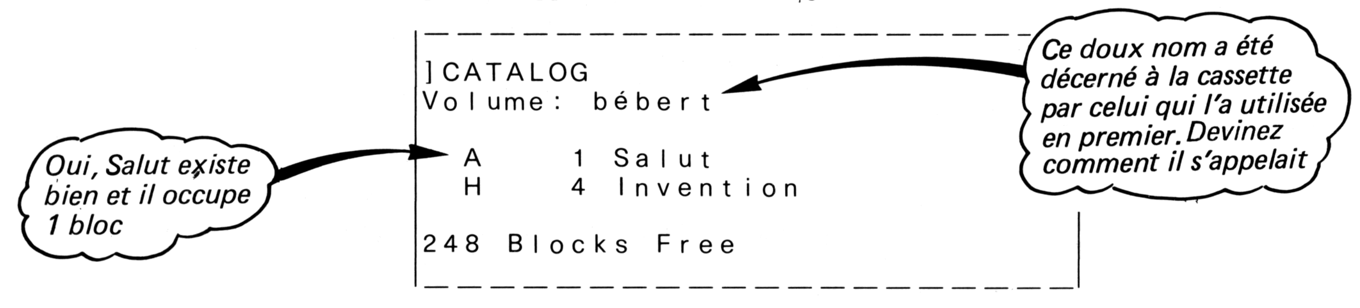
Il ne reste plus qu'à donner l'ordre (ne placez surtout pas de guillemets autour de *Salut) :*

SAUVEGARDE

SAVE

```
]SAVE Salut
```

Dès le RETURN, la cassette s'anime pendant quelques secondes, cherchant où elle va ranger ce programme, le faisant, inscrivant son nom dans le catalogue, etc. Dès qu'elle stoppe, vérifiez son catalogue :



Faites NEW pour effacer la mémoire, puis LIST afin de vérifier qu'elle ne contient plus rien.

2. Chargement

On va, maintenant, relire le programme *Salut* sur la cassette et recharger ce programme en mémoire centrale.

Cette opération s'appelle un chargement.
En anglais, cela se dit LOAD.

Frappez l'ordre (ne placez surtout pas de guillemets autour de *Salut*).
Dès le RETURN, la cassette s'anime, le programme est lu sur la cassette et transféré en mémoire centrale, ce qu'un LIST ou un RUN vous prouvera :

CHARGEMENT
LOAD

```
]NEW

]LIST

]LOAD Salut

]RUN
Bonjour

]LIST

   10 PRINT "Bonjour"
```

Il n'y a plus rien en mémoire

Ordre de chargement du programme

Attention, toutefois, à bien orthographier vos noms, car le Basic est très pointilleux sur ce chapitre.

Ainsi, une simple erreur de majuscule vous fera aboutir à un échec :

Il fallait écrire Salut avec un S majuscule

```
]LOAD salut

File Not Found
```

«Je n'ai pas trouvé ce fichier»

Par contre, vous pouvez vous épargner d'avoir à faire un LOAD Salut, puis un RUN, en court-circuitant le LOAD. Il vous suffira de frapper *RUN Salut :* le programme sera chargé et aussitôt exécuté :



3. Fichiers de sauvegarde

Si vous sauvegardez un fichier *avec le même nom* qu'un fichier préexistant, ce dernier sera remplacé par le nouveau. Toutefois, Adam, qui sait être vigilant pour deux, n'effacera pas l'ancien mais le placera « en réserve », supposant que, peut-être, vous serez saisi d'un remords. Vérifions-le en modifiant notre ligne 10 ainsi, puis en la sauvegardant :

```
   10 PRINT "Bonjour à tous"
]save Salut
```

Demandons le catalogue : il nous indique la présence de *deux* fichiers *Salut* mais attention, l'un est précédé d'un A majuscule et c'est le *vrai* fichier, le nouveau, alors que l'autre, l'ancien s'est vu affligé d'un petit *a :*

```
]CATALOG
Volume: bébert

  a     1 Salut
  H     4 Invention
  A     1 Salut

247 Blocks Free
```

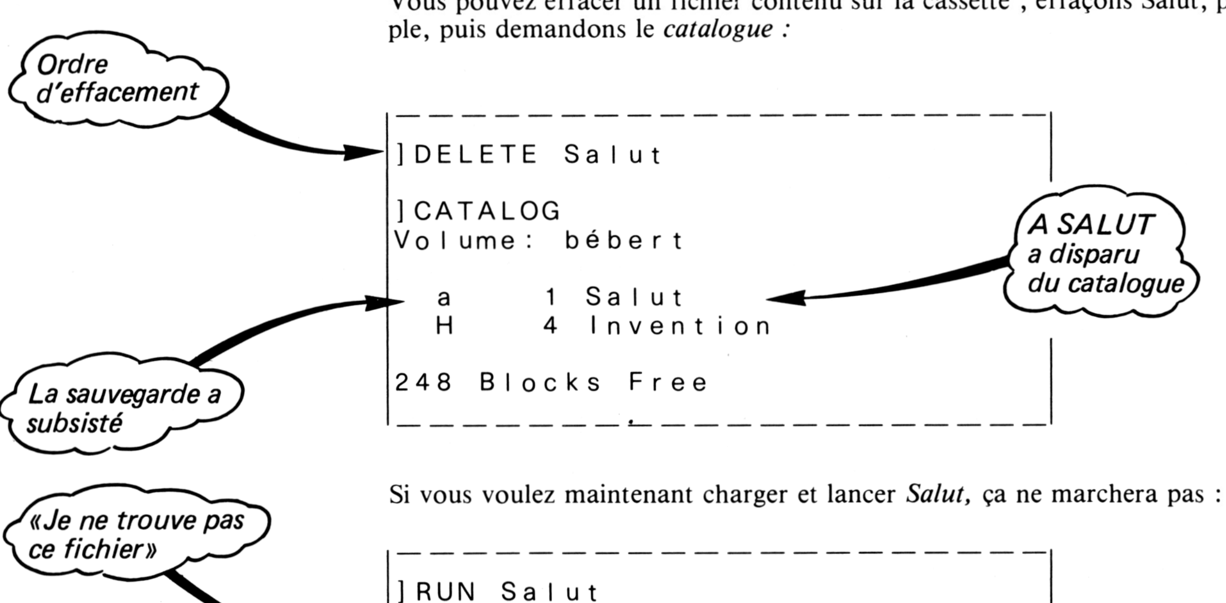
Si vous faites un *RUN Salut,* c'est le nouveau programme qui est chargé et lancé :

DELETE FICHIER

```
]RUN Salut
Bonjour a tous

]
```

Vous pouvez effacer un fichier contenu sur la cassette ; effaçons Salut, par exemple, puis demandons le *catalogue :*




```
]DELETE Salut

]CATALOG
Volume: bébert

  a     1 Salut
  H     4 Invention

248 Blocks Free
```

Si vous voulez maintenant charger et lancer *Salut,* ça ne marchera pas :

```
]RUN Salut

File Not Found
```

VERROUILLAGE FICHIER

LOCK

Peut-être voulez-vous protéger certains fichiers, interdire leur effacement accidentel ? Après les avoir sauvegardés, il faudra les « verrouiller » avec l'ordre LOCK *(« verrouiller »),* suivi du nom du fichier. Créons le programme *adieux* et verrouillons-le :

```
]10 PRINT "Au revoir"

]save adieux

]LOCK adieux

]catalog
Volume: bébert

  a     1 Salut
  H     4 Invention
 *A     1 adieux

247 Blocks Free
```

Sauvegarde du fichier adieux

Verrouillage en protection, du fichier adieux sur bande

Un astérisque : c'est la marque du verrouillage

Désormais, vous pouvez lire *adieux* et le lancer, mais vous ne pouvez plus l'effacer :

```
]DELETE adieux

File Not Found

]

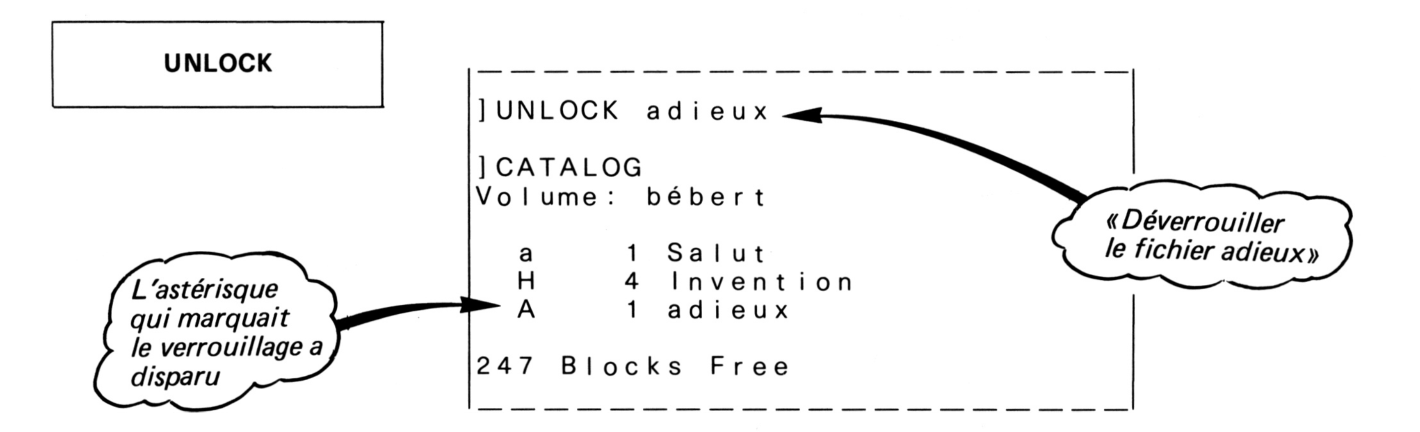
RUN adieux
Au revoir
```

«Je ne peux pas le trouver» ... parce qu'il est protégé

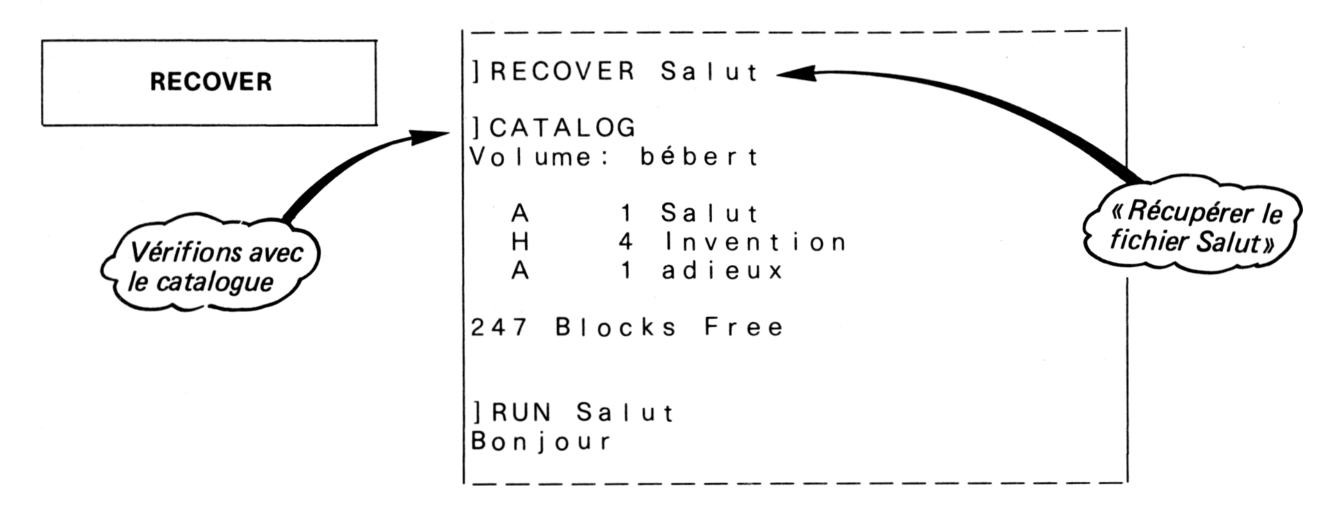
«Supprimer le fichier adieux»

Par contre, vous pouvez parfaitement le lire et le lancer

Si, vraiment, vous tenez à l'effacer (quelle inconstance), il vous faudra au préalable le déverrouiller avec UNLOCK :



Vous le traitez maintenant tel un fichier « normal ». A propos, peut-être voulez-vous récupérer le fichier de sauvegarde *Salut* qui se trouve toujours sur bande avec son petit *a ?* Cet ordre de récupération se dit RECOVER ; il va lui faire perdre sa qualité de *fichier de sauvegarde* et lui restituera à la fois celle de *fichier* et son A majuscule. Dès lors, vous pouvez l'exploiter :




```
]RECOVER Salut

]CATALOG
Volume: bébert

  A     1 Salut
  H     4 Invention
  A     1 adieux

247 Blocks Free


]RUN Salut
Bonjour
```

Voici donc les principales fonctions relatives à la sauvegarde et au chargement des programmes.

4. Pour rebaptiser un fichier

Changer le nom d'un fichier demande une manœuvre supplémentaire ; elle consiste à prévenir le Basic qu'on s'adresse à la cassette en frappant MON C. Commençons par examiner le catalogue de la cassette (entre temps, l'auteur a ajouté un nouveau fichier, un programme que vous trouverez plus loin) :

MON

```
]CATALOG
Volume: bébert

  A     1 Salut
  H     4 Invention
  A     1 adieux
  A     1 Lutins

246 Blocks Free
```

On va changer le nom de « Lutins » en « Sprites », son équivalent américain ; pour cela, frappez successivement :

RENAME

```
]MON C

]RENAME Lutins,Sprites
```

Au ↲, l'unité à cassette s'anime et le nom est modifié, ce que l'on peut vérifier avec :

```
]CATALOG
Volume: bébert

  A     1 Salut
  H     4 Invention
  A     1 adieux
  A     1 Sprites

246 Blocks Free
```

Pour terminer l'ordre MON C, il faut maintenant frapper son inverse :

NOMON

```
]NOMON  C
```

L'ordre MON offre encore d'autres possibilités. Par exemple, MON I affiche les données provenant de la cassette, MON O affiche celles allant vers la cassette, MON L expose ligne après ligne les données au cours d'un chargement ; dans ce dernier cas, on commanderait MON L puis LOAD *nom du fichier*. Il faudra toujours « refermer » MON avec un NOMON suivi des mêmes ordres (NOMON L, par exemple).

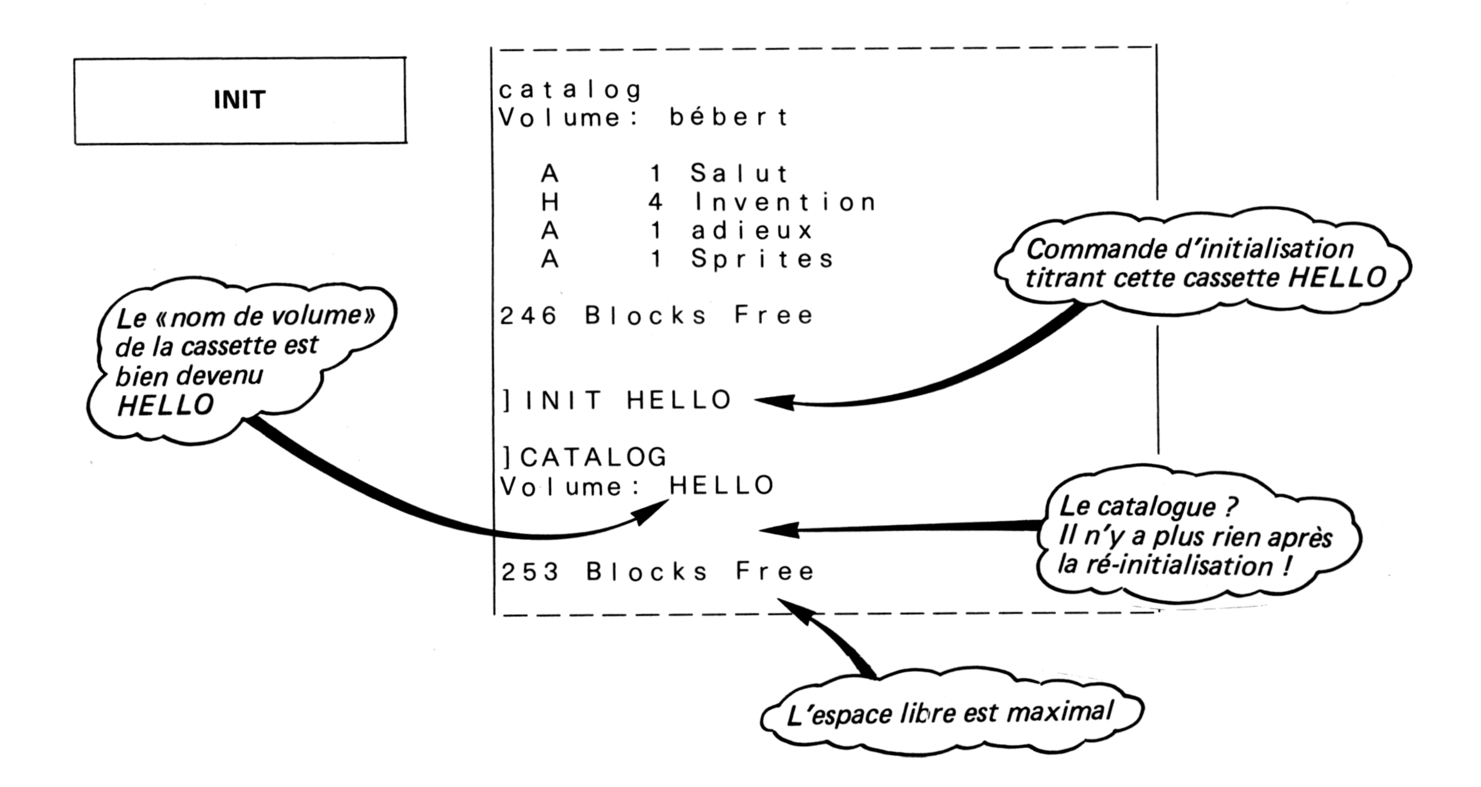
Le nom de cette cassette ne fait pas très sérieux : *bébert !* Revenons à un nom plus classique, que les informaticiens décernent bien volontiers et qui pèche par manque d'originalité : HELLO. Il existe une commande qui sert tout à la fois à *initialiser* une cassette, à lui attribuer un nom, mais aussi à effacer son contenu pour autant que vous ayez déjà enregistré des fichiers ; c'est INIT. *Donc, ne l'utilisez qu'avec d'infinies précautions.* Une séquence complète de démonstration est la suivante :

1) On demande le catalogue de la cassette.

2) On commande INIT *suivi du nom* à décerner à cette cassette.

3) Après le RETURN, la cassette s'anime un instant.

4) Dès que cette initialisation est terminée, on redemande le catalogue : en principe, il doit être vide car tout a été effacé. La preuve :



Un autre conseil : sauvegardez vos séquences importantes sur *deux* cassettes différentes au lieu d'une. En cours de travail, en outre, exécutez des sauvegardes régulières, tous les quarts d'heure, par exemple.

Questions sur le chapitre VIII

Nous ne vous proposons que deux questions : fondamentales !

1. Pour faire apparaître les noms de fichiers sur une cassette, il faut frapper :
   ☐ DIRECTORY
   ☐ CATALOG

2. L'ordre de sauvegarde sur cassette se dit :
   ☐ LOAD
   ☐ SAVE

Réponses page 147

Réponses aux questions sur le chapitre VI

1. Pour obtenir une valeur aléatoire, il faut placer un argument entre parenthèses et après RND. On écrira donc ? RND(1985). L'argument positif (et non nul) n'a aucune attribution ; mettez ce que vous voulez.

2. INT(3.14) donnera l'entier, donc 3.

3. Avec PRINT INT(RND(9) * 10) + 1 vous obtiendrez des valeurs entre 1 et 10.

4. Vous affichez *Au revoir* car la condition est vraie (5 est supérieur à 3), donc on va en 30.

5. Vous afficherez 0 car l'égalité est fausse.

6. L'égalité est fausse, mais NOT inverse le résultat, donc le Basic affichera 1.

GRAPHIQUE HAUTE DÉFINITION

On va examiner comment manipuler des données avec deux nouveaux ordres, DATA et READ. En application, on va étudier comment on réalise des « lutins », ou modules graphiques (que les Américains appellent « sprites »). Mais attention : leur conception n'est pas dépourvue de difficulté. L'essentiel était de vous prévenir !

Principaux thèmes étudiés

DATA	Sprites
READ	Modules graphiques
RESTORE	Tracé d'un lutin
Graphique haute définition	Binaire
HGR	Décimal
Grille graphique HGR	POKE
Couleurs en HGR	DRAW
HPLOT	SCALE
HCOLOR	ROT
Tracés en HGR	XDRAW
Rectangle en HGR	Vidéo inverse
HGR2	INVERSE
Ciel étoile	NORMAL
Rectangles emboîtés	FLASH
Lutins	

1. L'introduction de données par DATA et READ

DATA

READ

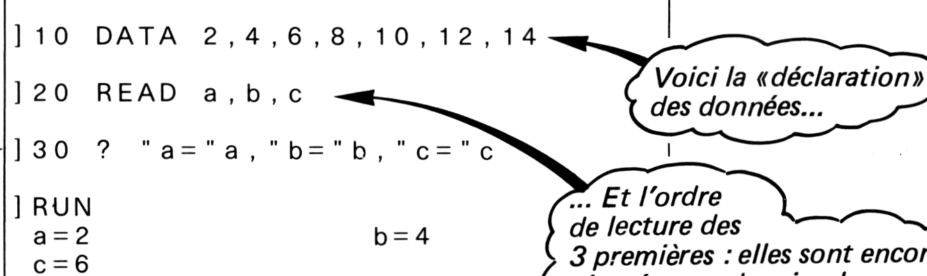
Chaque fois qu'un programme a eu besoin de données on a, jusqu'à présent :

1. — Soit inclus ces données *dans* le programme même, directement.
2. — Soit utilisé un INPUT qui les demandait à l'opérateur.

Or, il existe une troisième façon de procéder et elle consiste à lister ces données dans des lignes séparées du programme, précédées de la commande DATA (ce qui signifie *données) ;* lorsque le programme en aura besoin, on lui ordonnera de les lire dans l'ordre avec la commande READ (« lire »). En voici un premier exemple : on introduit, avec *data,* une liste de 7 nombres *(séparés par des virgules)* puis on commande leur lecture et leur affectation à 3 variables, *a, b* et *c.* La machine va alors lire les trois premiers et les affecter aux variables :

Les données doivent être séparées par des virgules

Ici, dans l'ordre PRINT, les virgules commandent la tabulation.



Dans DATA et READ, les variables doivent être séparées par des virgules (elles ne constituent pas un ordre de tabulation : ce n'est qu'avec PRINT que la virgule sert à tabuler).

Remarquez également qu'on a directement placé le nom de la variable après la formule à afficher, par exemple "*a* = "*a*, ce qui est le plus simple (puisque le Basic le permet !).
On peut d'ailleurs fort bien poursuivre et demander la lecture de deux autres données. Dans ce cas, l'ordinateur a conservé en mémoire le fait que les trois premières avaient déjà été lues, aussi prendra-t-il les deux suivantes :

```
]40 READ i,j

]50 ? "i="i,"j="j

]RUN
 a=2                    b=4
 c=6
 i=8                    j=10
```

Allons jusqu'au bout de la liste, et dépassons-la même *(« lorsque les bornes sont franchies, il n'y a plus de limites » :* pensée profonde de Pierre Dac) :

On demande 4 lectures alors qu'il ne reste plus que trois nombres... Comment l'ordinateur va-t-il s'en sortir ?

```
]60 READ r,s,t,u

]70 ? "r="r,"s="s,"t="t,"u="u

]RUN
 a=2                    b=4
 c=6
 i=8                    j=10

?Out of DATA Error In 60
```

En vous décochant ce message d'erreur : «Plus de données en 60»

On aurait pu aussi exécuter la lecture à l'aide d'une boucle, telle que la suivante :

```
]10 DATA 2,4,6,8,10,12,14

]20 FOR n=1 TO 4

]30 READ n: ? n,

]40 NEXT

]RUN
 2                      4
 6                      8
```

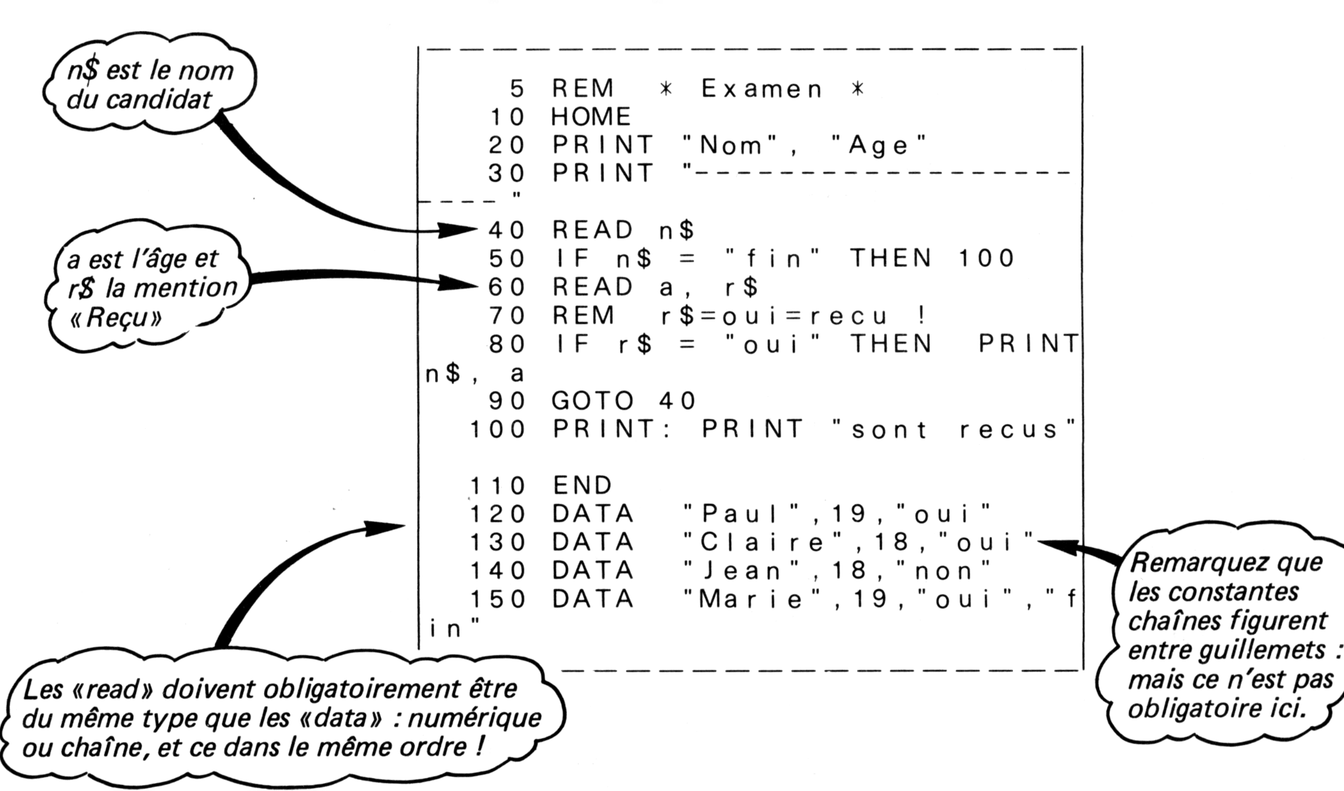
Les données de *data* peuvent être réparties sur plusieurs lignes sans que cela change quoi que ce soit : l'ordinateur les lira toujours dans l'ordre où elles se présentent.

En outre, ces déclarations de données (DATA) peuvent être placées n'importe où dans un programme, même après la respectable instruction END qui indique sa fin.

En voici un exemple :

```
]10 FOR n=1 TO 7
]20 READ d: ? d,
]30 NEXT
]40 END
]50 DATA 2,4,6
]60 DATA 8,10,12,14
]RUN
 2                    4
 6                    8
 10                   12
 14
```

Une application typique des ordres *data* et *read* concerne les fichiers ; voici un programme qui liste les candidats reçus à un examen et qui recourt à des variables chaînes et des variables numériques (c'est là son principal mérite !). Remarquez bien la correspondance obligatoire entre le nom de variable et ses constantes :



```
    5 REM  * Examen *
   10 HOME
   20 PRINT "Nom", "Age"
   30 PRINT "------------------
----"
   40 READ n$
   50 IF n$ = "fin" THEN 100
   60 READ a, r$
   70 REM r$=oui=recu !
   80 IF r$ = "oui" THEN PRINT
n$, a
   90 GOTO 40
  100 PRINT: PRINT "sont recus"

  110 END
  120 DATA "Paul",19,"oui"
  130 DATA "Claire",18,"oui"
  140 DATA "Jean",18,"non"
  150 DATA "Marie",19,"oui","f
in"
```

Notez que les déclarations des constantes chaînes ne doivent pas figurer obligatoirement entre guillemets, comme d'habitude avec ce type de constantes. On ne mettra généralement des guillemets que si elles contiennent des virgules, points-virgules, ou espaces en préfixe ou suffixe.

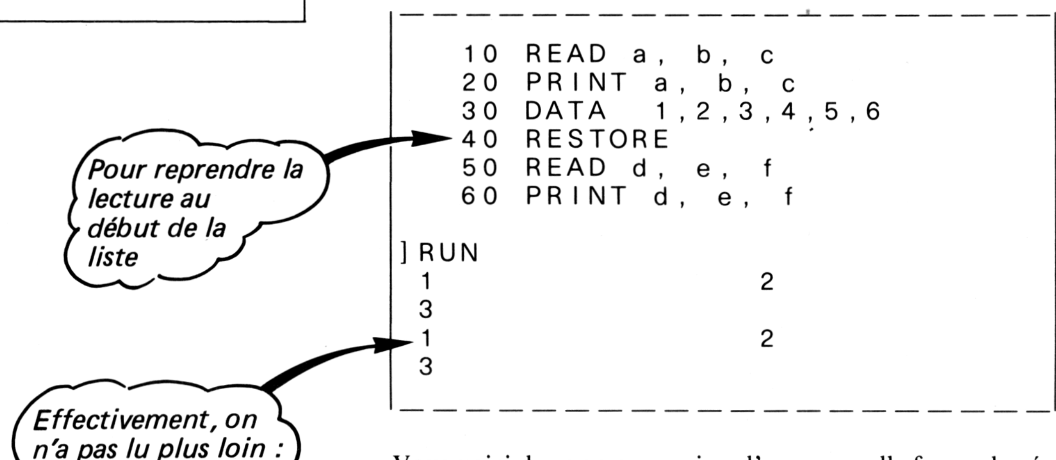
Au *run,* l'écran s'efface et vous voyez apparaître :

```
Nom                    Age
----------------------
Paul                   19
Claire                 18
Marie                  19

sont recus
```



Lorsqu'une liste de données est épuisée, on peut la reprendre à son début en plaçant l'ordre RESTORE *(restaurer).* En voici un exemple :




```
    10 READ a, b, c
    20 PRINT a, b, c
    30 DATA  1,2,3,4,5,6
    40 RESTORE
    50 READ d, e, f
    60 PRINT d, e, f

]RUN
 1                 2
 3
 1                 2
 3
```

Vous voici donc en possession d'une nouvelle façon de gérer les données qui va aussitôt nous servir.

2. Graphique haute résolution

GRAPHIQUE HAUTE DEFINITION
HGR

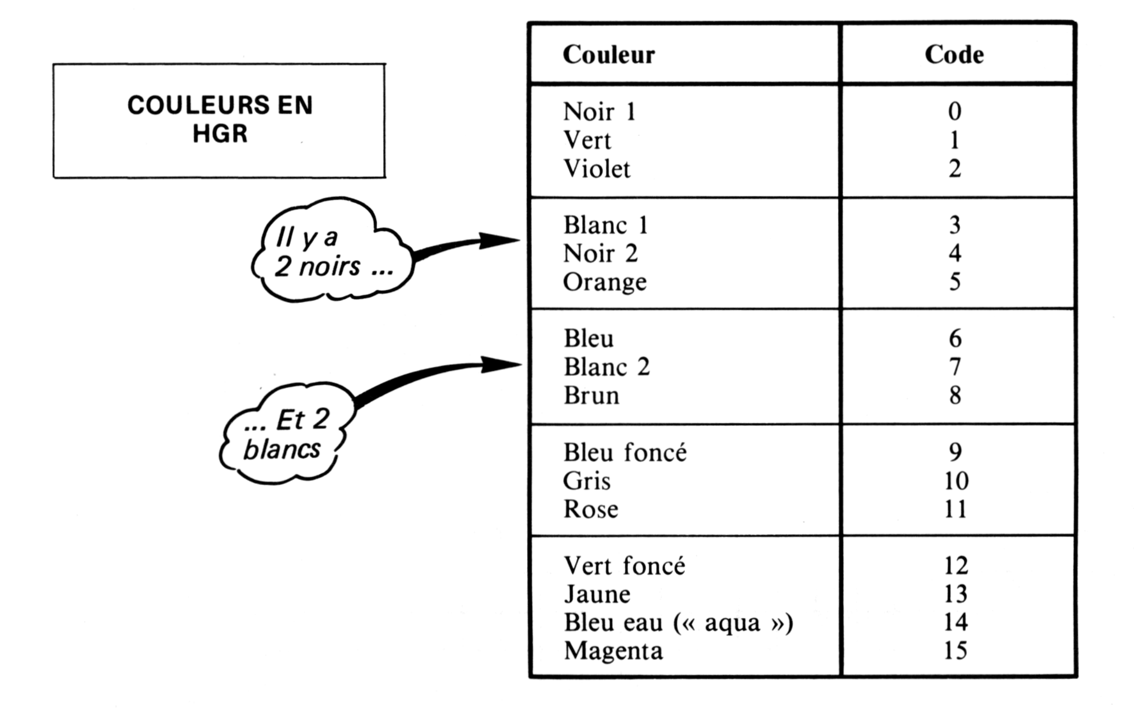
Pour marquer une pause (si l'on peut dire), repassons aux dessins, mais cette fois en graphique *haute définition.* Rappelez-vous qu'en *basse définition graphique,* obtenue avec GR, l'écran se divisait en 40 rangées de 40 colonnes. Ici, en *graphique à haute définition,* il va être de 160 lignes de 256 colonnes : c'est dire que chaque point élémentaire sera très fin, ce qui va permettre une grande finesse de tracé.

En mode à haute définition, on va retrouver des ordres semblables à ceux employés en basse définition mais précédés de la lettre H.

Ainsi, on va trouver :

- HGR au lieu de GR pour passer en haute définition,
- HCOLOR au lieu de COLOR, pour définir une couleur,
- HPLOT pour allumer un point (au lieu de PLOT).

Mais attention, dans HLIN et HTAB, le H vient pour « horizontal » et HLIN ne fonctionne pas en HGR, non plus que VLIN. Notez aussi que les 16 couleurs sont différentes en HGR et répondent à d'autres codes qui sont :



Couleur	Code
Noir 1	0
Vert	1
Violet	2
Blanc 1	3
Noir 2	4
Orange	5
Bleu	6
Blanc 2	7
Brun	8
Bleu foncé	9
Gris	10
Rose	11
Vert foncé	12
Jaune	13
Bleu eau (« aqua »)	14
Magenta	15

HPLOT

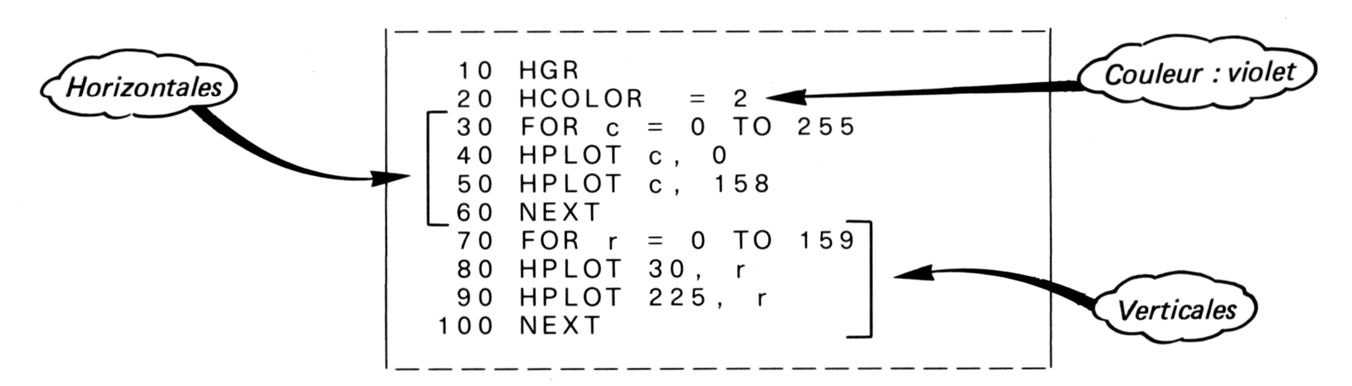
HCOLOR

Pour expérimenter cet écran HGR, on va tracer non pas un cadre le circonscrivant (si l'image, sur votre téléviseur, est un peu décalée à gauche ou à droite, vous ne verriez pas les montants verticaux) mais des lignes horizontales l'encadrant et des verticales un peu décalées (de 30 lignes) vers l'intérieur. Notez que la syntaxe de HPLOT est semblable à celle de PLOT :

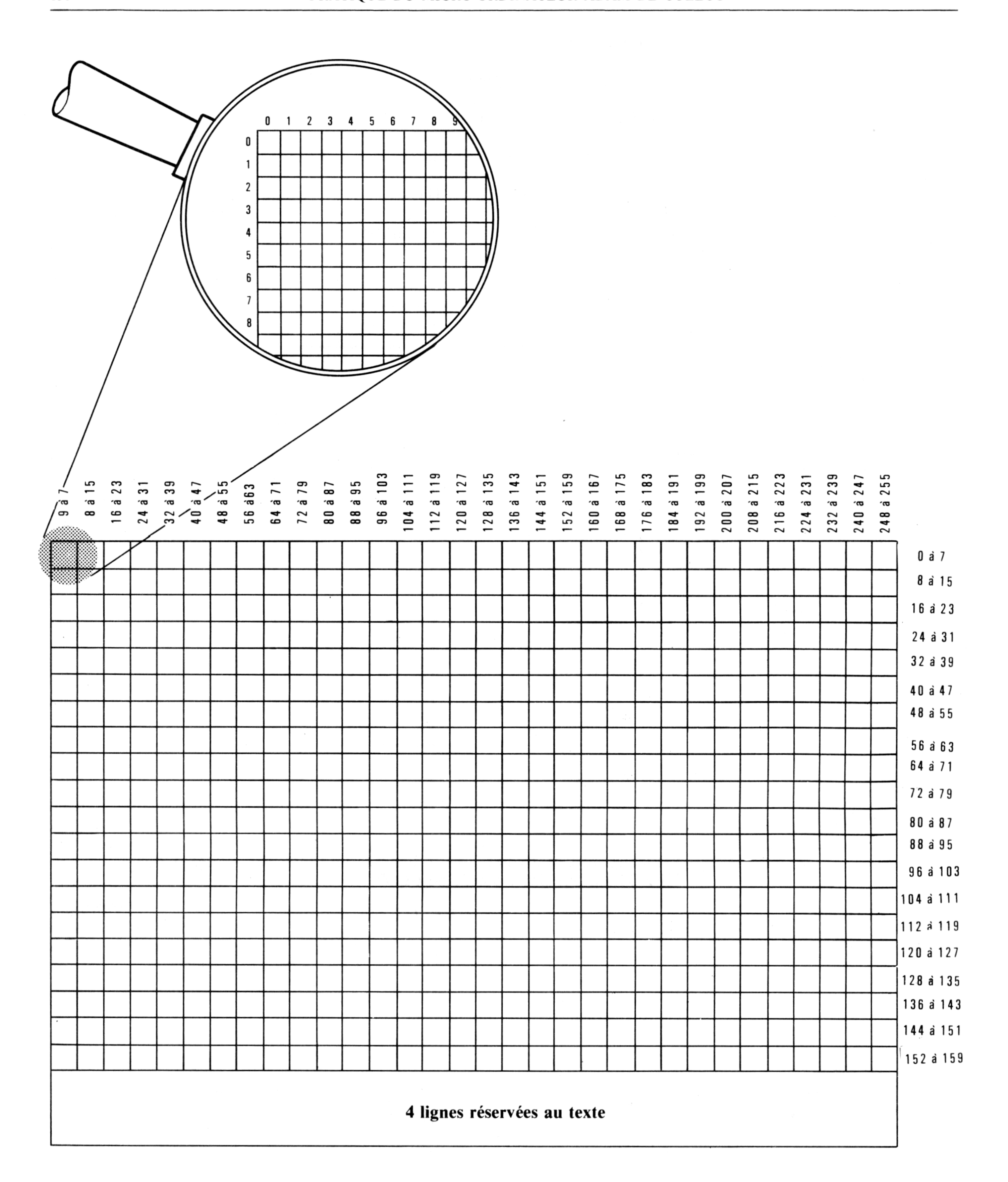
HPLOT *colonne,rangée*

Virgule obligatoire

Entrez ce programme et lancez-le. Il va faire apparaître, en violet, une sorte de cadre au tracé très fin, que nous ne vous représentons pas ici :




```
 10 HGR
 20 HCOLOR  = 2
 30 FOR c = 0 TO 255
 40 HPLOT c, 0
 50 HPLOT c, 158
 60 NEXT
 70 FOR r = 0 TO 159
 80 HPLOT 30, r
 90 HPLOT 225, r
100 NEXT
```

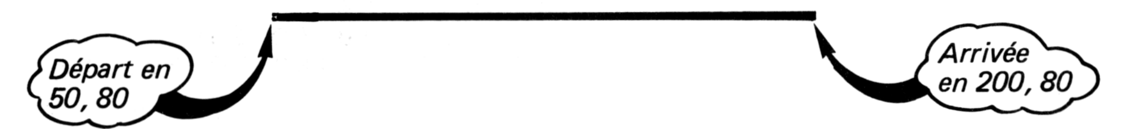


Organisation de l'écran graphique haute définition
256 × 160 points

Pour revenir à l'écran en mode texte, tappez TEXT puis RETURN (rappelons que c'est le ↲), comme pour GR. Ajoutons à ce « cadre » une diagonale, en vert, ce qui donne :

```
 10 HGR
 20 HCOLOR = 2
 30 FOR c = 0 TO 255
 40 HPLOT c, 0
 50 HPLOT c, 158
 60 NEXT
 70 FOR r = 0 TO 159
 80 HPLOT 30, r
 90 HPLOT 225, r
100 NEXT
105 HCOLOR = 1
110 p = 0
120 HPLOT p, p
130 p = p+1
140 IF p = 159 THEN END
150 GOTO 120
```

Pour revenir à l'écran de texte, frappez TEXT puis RETURN.
Cependant, il existe en HGR une méthode plus rapide et plus élégante pour tracer des droites : il suffit de donner au Basic les coordonnées des points de départ et d'arrivée. Ainsi, soit à tracer la droite :



On commandera simplement un HPLOT 50, 80 TO *(ce qui signifie « vers »)* 200, 80. Voici le programme que vous pouvez aussitôt lancer :

TRACES EN HGR

```
20 HGR
30 HCOLOR = 11
40 HPLOT 50, 80 TO 200, 80
```

Rose

Tracé d'une droite

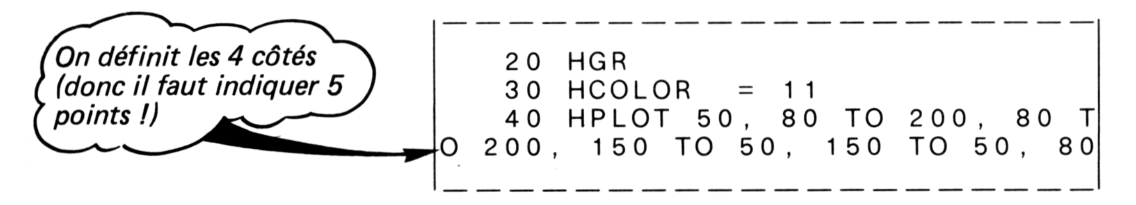
RECTANGLE EN HRG

Il fournit une belle droite, en rose (couleur = 11). Mais on peut encore faire mieux et considérer le point d'arrivée comme un nouveau point de départ. Il suffit de prolonger la formule avec des TO *(vers)*. Par exemple, et pour tracer ce rectangle rose :

50,80 200,80

50,15 200,150

Rédigez le programme suivant et lancez-le :



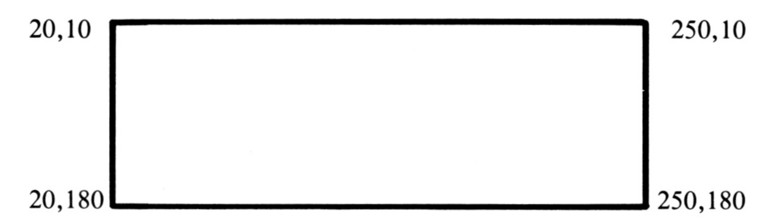

```
20 HGR
30 HCOLOR = 11
40 HPLOT 50, 80 TO 200, 80 T
O 200, 150 TO 50, 150 TO 50, 80
```

Bien, maintenant que vous maîtrisez ce fonctionnement, passons à un « grand écran » : on va supprimer les 4 lignes de texte en bas et les remplacer par la suite de l'écran graphique haute résolution. Cette fois, on dispose de 192 lignes numérotées de 0 à 191, sur 256 colonnes numérotées de 0 à 255. Ce nouvel HGR s'appelle HGR2 et fonctionne de la même façon.

HGR2

Attention, toutefois, car vous allez travailler à l'aveuglette pour le texte : vous pourrez frapper des ordres acquis par la machine mais vous ne les verrez pas s'afficher.

Sur le même principe que ci-dessus, on va tracer un rectangle dans ce beau bleu appelé AQUA, couleur 14 :



Entrez ce programme en machine et lancez-le pour l'obtenir :

```
10 HGR2
20 HCOLOR = 14
30 HPLOT 20, 10 TO 250, 10 TO 250, 180 TO 20, 180 TO 20, 10
```

Pour revenir en mode texte, frappez (sans rien voir) TEXT au clavier, puis faites RETURN.

Essayez encore maintenant de remplir l'écran de façon aléatoire, avec des points aléatoires, ainsi qu'on l'avait fait sous le nom de « mosaïque » en basse définition :

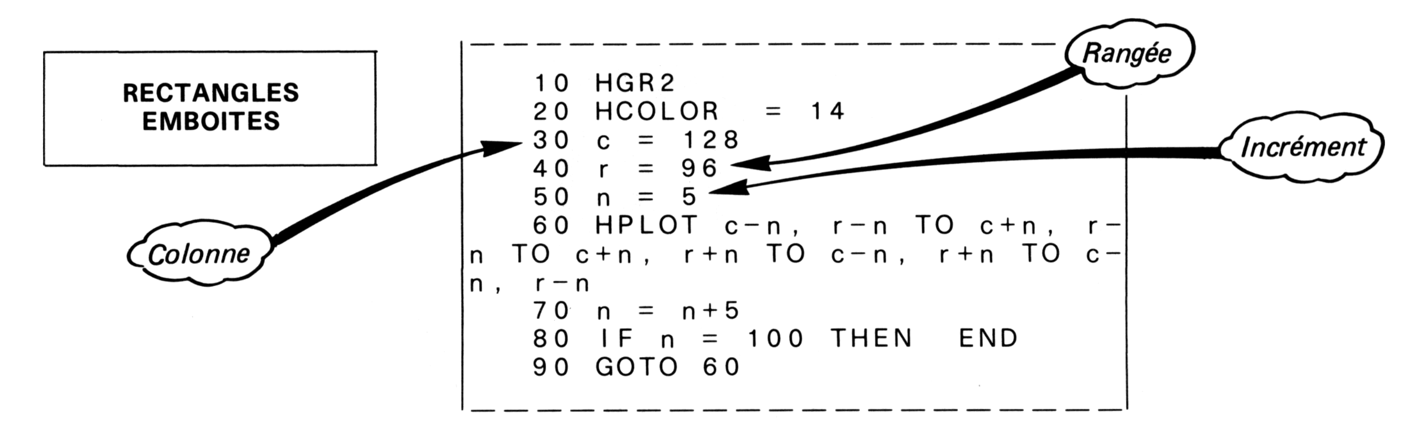
CIEL ETOILE

```
 5 REM  mosaique en hgr2
10 HGR2
30 t = RND(9)*15        <- Teinte aléatoire
40 r = RND(9)*192
40 c = RND(9)*256
60 HCOLOR = t
70 HPLOT c, r
80 GOTO 30
```

Lancez ce programme et cette fois, c'est vraiment un ciel constellé d'étoiles multicolores qui va se former sur votre écran.

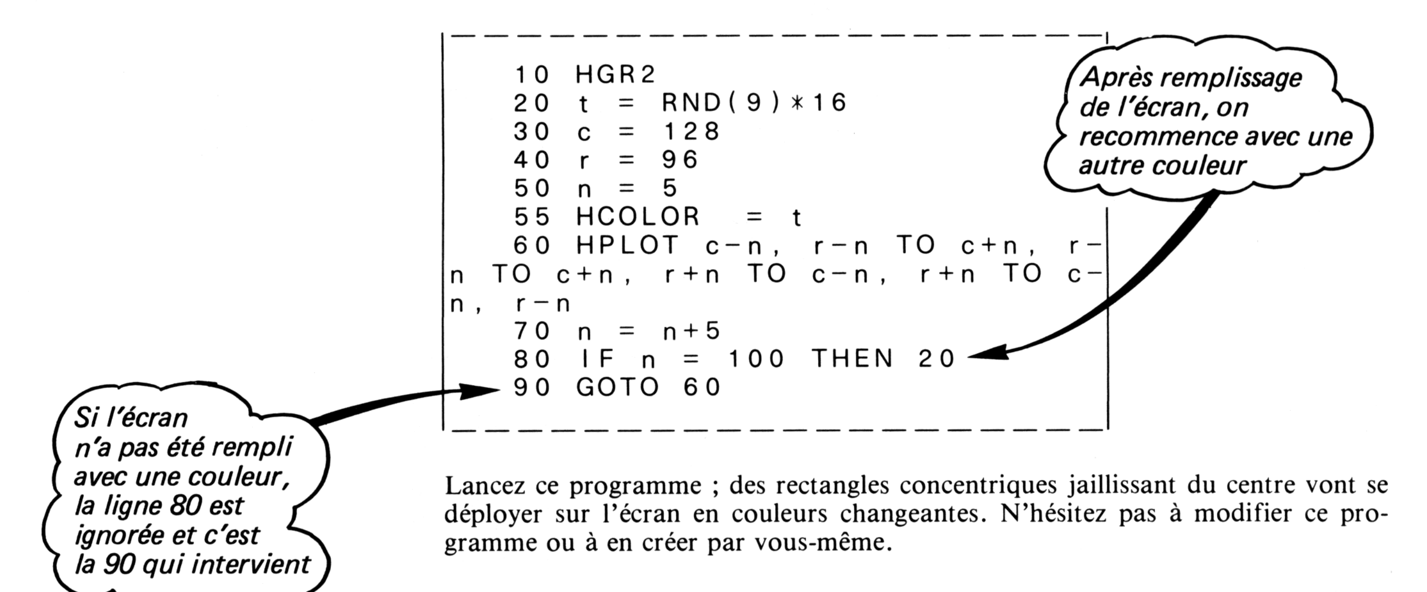
Remarquez qu'ici, on s'est à nouveau dispensé de prendre l'entier des valeurs aléatoires. Puisque le Basic veut bien le faire de lui-même avec bonne volonté !

Voulez-vous, maintenant, afficher des rectangles emboîtés ? Voici ce programme, où nous avons pris des variables pour simplifier l'écriture de la ligne 60 ; la variable *n* dont dépendent les dimensions du rectangle s'accroît par pas de 5 à chaque tour de boucle. La ligne 80 place une fin de programme lorsque l'écran est rempli :




```
10 HGR2
20 HCOLOR = 14
30 c = 128
40 r = 96
50 n = 5
60 HPLOT c-n, r-n TO c+n, r-n TO c+n, r+n TO c-n, r+n TO c-n, r-n
70 n = n+5
80 IF n = 100 THEN END
90 GOTO 60
```

Frappez TEXT puis RETURN afin de revenir au mode texte. Voulez-vous donner du « mouvement » à ces dessins concentriques en les reprenant à chaque fois dans une couleur différente ? Tirez-la au hasard :

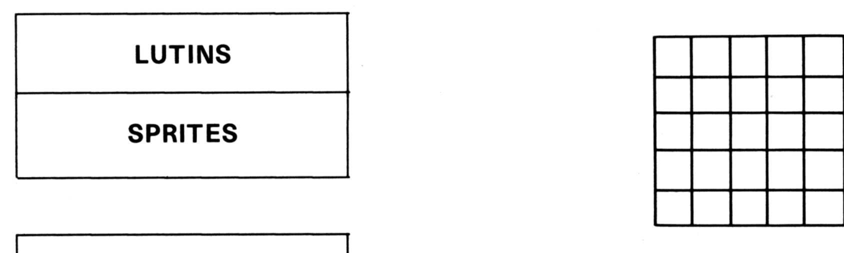



```
10 HGR2
20 t = RND(9)*16
30 c = 128
40 r = 96
50 n = 5
55 HCOLOR = t
60 HPLOT c-n, r-n TO c+n, r-n TO c+n, r+n TO c-n, r+n TO c-n, r-n
70 n = n+5
80 IF n = 100 THEN 20
90 GOTO 60
```

Lancez ce programme ; des rectangles concentriques jaillissant du centre vont se déployer sur l'écran en couleurs changeantes. N'hésitez pas à modifier ce programme ou à en créer par vous-même.

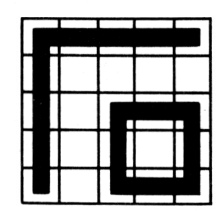
3. Création de modules graphiques

Votre Adam vous autorise la création de formes qu'on peut ensuite commander en bloc. C'est ce que les Américains appellent des « *sprites* », littéralement des *lutins,* mais qu'on pourrait aussi désigner par « modules graphiques ». C'est assez compliqué à réaliser pour un débutant, mais nous nous en voudrions d'escamoter cette initiation. Sachez, tout d'abord, qu'un lutin s'inscrit dans un quadrillage graphique haute résolution, par exemple de 5 lignes de 5 colonnes :



MODULES GRAPHIQUES

Il faut d'abord inscrire dans ce pavé le tracé à réaliser. Nous vous proposons ce tracé bizarre auquel nous n'attribuons aucun sens particulier :



Pour obtenir ce tracé, on va placer des segments de droite qui se prolongeront pour les traits continus. C'est ce qu'on a dessiné ci-après ; les flèches continues indiquent *un tracé et le sens du mouvement ;* les discontinues indiquent un déplacement au cours duquel on a « levé le crayon ».

La description d'un « lutin » exige, en effet, qu'on suive un chemin continu, quel qu'il soit.

Nous avons choisi un chemin mais vous pourriez en adopter un autre, il n'y a là pas de règles. Nous avons numéroté les segments ou les déplacements afin que vous puissiez mieux vous y retrouver par la suite :

TRACE D'UN " LUTIN "

Maintenant, il faut coder tout cela, selon les codes suivants :

Déplacement seul (« crayon levé »)		**Déplacement tracé**	
Sens	Code	Sens	Code
⇡	000	↑	100
⇢	001	→	101
⇣	010	↓	110
⇠	011	←	111

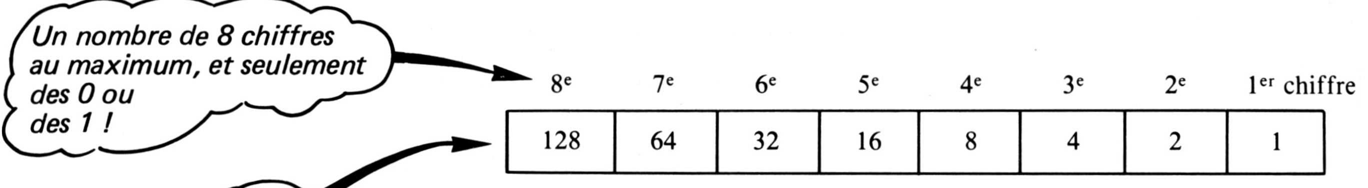
En suivant notre tracé, on va maintenant coder le circuit ; cela donne :

Segment	Code
1 2	111 111
3 4	111 111
5 6	110 110
7 8	110 110
9 10	001 001
11 12	101 101
13 14	100 100
15 16	111 111
17 18	110 110

Ce n'est pas tout : il faut assembler ces codes par deux ou par trois, à la condition de ne pas dépasser 8 chiffres ; si le neuvième, celui de gauche, est un zéro, on le supprime. Cela donne :

Code	Pour les segments
111111	2 et 1
111111	4 et 3
110110	6 et 5
01110110	9 et 8 et 7
101001	11 et 10
101101	13 et 12
111100	15 et 14
110111	17 et 16
110	18

Vous tenez toujours bon ? Si oui, sachez que tous ces *zéros* et ces *uns* s'appellent du *binaire* et qu'il va falloir traduire cela en décimal ! Nous n'allons vous apprendre du binaire que les règles de conversion en décimal. Partons d'un mot de 8 chiffres binaires (des 0 ou des 1) ; chaque chiffre a une valeur décimale précise en fonction de sa position ; cette valeur est de :



Valeur du chiffre

Le règle est facile : on place le nombre sur cette grille, à partir de la droite (s'il manque des chiffres, on suppose que ce sont des zéros *à gauche),* puis, chaque fois qu'il y a un 1, on relève la valeur correspondante (ce qu'on appelle *le poids),* et on fait l'addition de tous ces poids. Par exemple :

- 11111111 (des 1 partout) sont équivalents à 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255
- 00000000 (des zéros partout) font... zéro
- 00000011 valent 2 + 1 = 3 (de même que 0011, par exemple)
- etc.

BINAIRE

DECIMAL

A partir de cette règle, transposons nos codes binaires. On obtient, dans le même ordre que précédemment :

Code binaire	Equivalent décimal
111111 111111	63 63
110110 01110110	54 118
101001 101101	41 37
111100 110111	60 55
110	6

Vous pouvez pousser un soupir de soulagement, les calculs sont terminés ! On va introduire ces valeurs dans un programme qu'on ne vous commentera guère car il exigerait encore bien davantage de pages ; il vous permettra toutefois de comprendre le fonctionnement des « lutins ». Notez cependant que :

- La ligne 30 définit la zone mémoire où l'on va ranger ce programme.
- La ligne 40 indique successivement dans ses données :
  — le numéro d'ordre du lutin, 01 ici, car il peut y en avoir plusieurs,
  — trois index pour la définition du lutin.
- Les lignes 50 et 60 reprennent les valeurs calculées ci-dessus.
- On y ajoute une donnée 00 qui indique au Basic que la description du « sprite » est terminée.
- Puis on va charger les positions mémoires successives, à partir de 51456 (ligne 80) avec les DATA lues par la ligne 70.

POKE

Pour charger une valeur dans une cellule mémoire précise dont on donne l'adresse, on utilise l'ordre POKE.

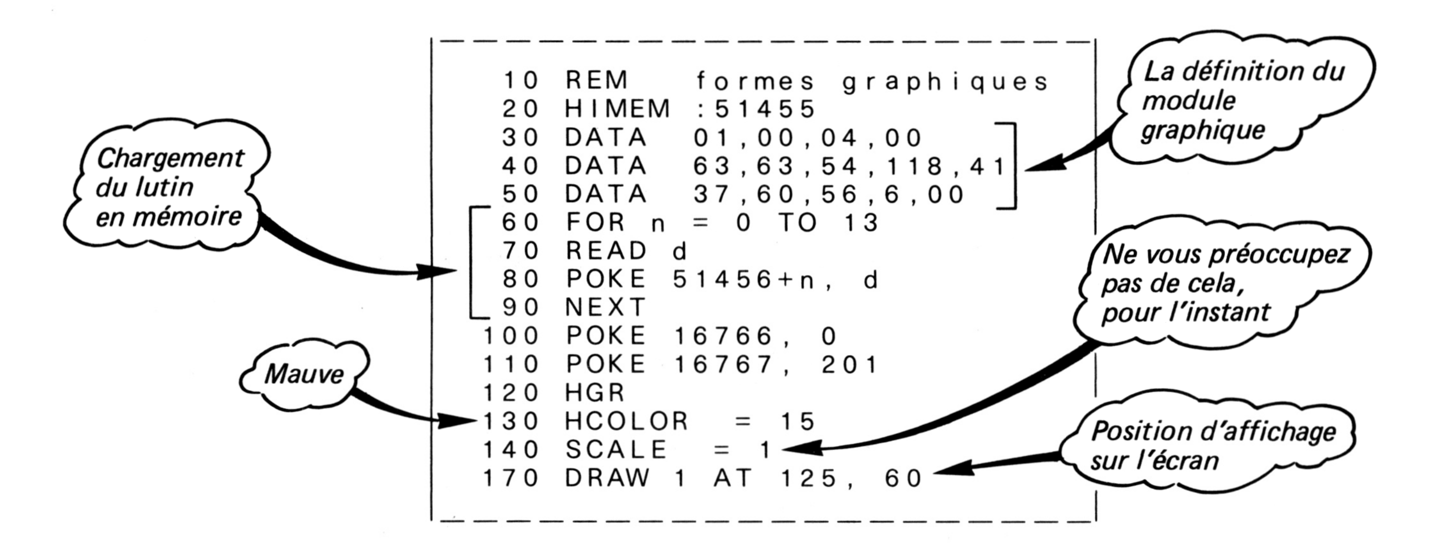
• D'autres POKE, en 100 et 110, indiquent au Basic où la définition du lutin est logée en mémoire.

• Enfin, le dessin est commandé à l'aide de l'ordre DRAW, *dessiner*.

DRAW

DRAW s'applique uniquement aux « lutins », en HGR ou HGR2. Il doit être suivi du numéro du lutin et de la position à laquelle le dessin apparaîtra sur l'écran.

Voici le programme :




```
 10 REM    formes graphiques
 20 HIMEM  :51455
 30 DATA   01,00,04,00
 40 DATA   63,63,54,118,41
 50 DATA   37,60,56,6,00
 60 FOR n = 0 TO 13
 70 READ d
 80 POKE 51456+n, d
 90 NEXT
100 POKE 16766, 0
110 POKE 16767, 201
120 HGR
130 HCOLOR  = 15
140 SCALE   = 1
170 DRAW 1 AT 125, 60
```

Lancez ce programme et vous verrez apparaître le module graphique dessiné à peu près au milieu de l'écran.

4. Mise en mouvement

Ce mode reste réellement bien petit ; grandeur : 1/1. Or, on peut obtenir un effet de loupe (ou de zoom) grâce à l'ordre SCALE qu'on avait introduit en ligne 140, en lui affectant le coefficient 1 pour obtenir l'échelle 1/1. Mais vous pouvez choisir d'autres coefficients, par exemple 8. Modifiez donc aussi la ligne 140 :

SCALE

```
]140 SCALE=8
```

Puis, relancez ce programme. Effectivement, le petit module a bien grandi. Essayez d'autres valeurs, pour voir.
Peut-être également son orientation ne vous convient-elle pas ? Qu'à cela ne tienne : on va le faire pivoter grâce à l'ordre ROT (pour rotation). On va même le faire tourner sur lui-même ; la rotation totale se commande avec ROT = 64, ce qui signifie que ROT = 32 commande une rotation de 180° et ROT = 16 de 90°.

ROT

Notez que ROT tout comme SCALE, sont généraux à HGR et ne s'appliquent pas seulement aux lutins.

Ajoutez cette séquence :

```
]150 FOR a=1 TO 64
]160 ROT=a
]190 NEXT
```

Puis lancez ce programme : effectivement, la forme graphique apparaît et tourne, ce qui donne le même effet que plusieurs vues superposées d'un mouvement sur un film. Afin de supprimer cette superposition, il faut *effacer* l'image précédente *avant* de créer la suivante ; pour cela, on dispose de l'ordre XDRAW, symétrique de DRAW, mais qui utilise la couleur complémentaire à HCOLOR en l'apportant sur le même dessin : ce qui revient à l'effacer. Ajoutez donc :

XDRAW

```
]180 XDRAW 1 AT 125,60
```

Le programme complet est désormais le suivant :

```
 10 REM     avec les lutins
 20 HIMEM :51455
 30 DATA  01,00,04,00
 40 DATA  63,63,54,118,41
 50 DATA  37,60,55,6,00
 60 FOR n = 0 TO 13
 70 READ d
 80 POKE 51456+n, d
 90 NEXT
100 POKE 16766, 0
110 POKE 16767, 201
120 HGR
130 HCOLOR = 15
140 SCALE = 8
150 FOR a = 1 TO 64
160 ROT = a
170 DRAW 1 AT 125, 60
180 XDRAW 1 AT 125, 60
190 NEXT
```

Cette fois et au RUN, c'est un véritable objet volant non identifié (OVNI) qui entre en scène, opérant une rotation complète sur fond de nuit noire. Cette rotation est d'autant plus lente que l'objet est grand (avec SCALE). Essayez aussi de déplacer son centre en jouant des lignes 170 et 180.
Nous vous concédons que la réalisation de ces modules n'est pas d'une simplicité biblique ; et encore ne vous a-t-on ni tout expliqué, ni tout dit. Que cela ne vous empêche cependant pas de programmer à votre tour des modules graphiques en reprenant ce même programme de base ; introduisez la description de vos modules dans les lignes 40 et 50 sans oublier de la terminer par un double zéro.

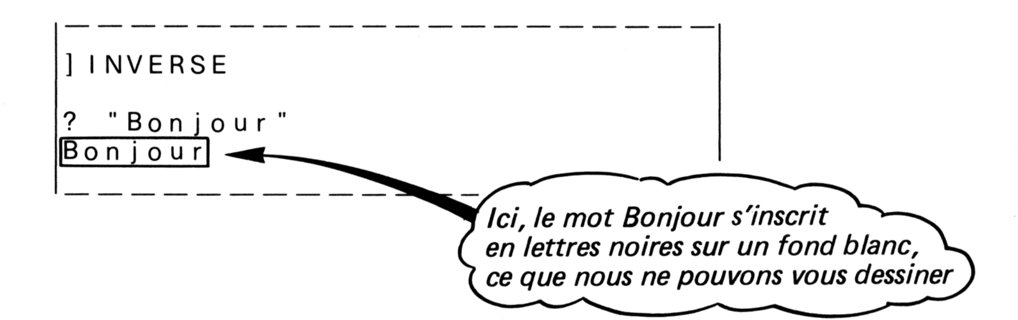
5. Vidéo inverse

Examinons pour terminer une fonction intéressante mais en mode TEXT uniquement, celle qui permet d'afficher du texte sur l'écran non plus en *blanc sur fond noir* mais en *noir sur fond blanc,* ce qu'on appelle la *vidéo inverse,* d'où l'ordre INVERSE. C'est facile à essayer en mode direct ; frappez d'abord INVERSE pour entrer dans ce mode, puis essayez :

VIDEO INVERSE

INVERSE

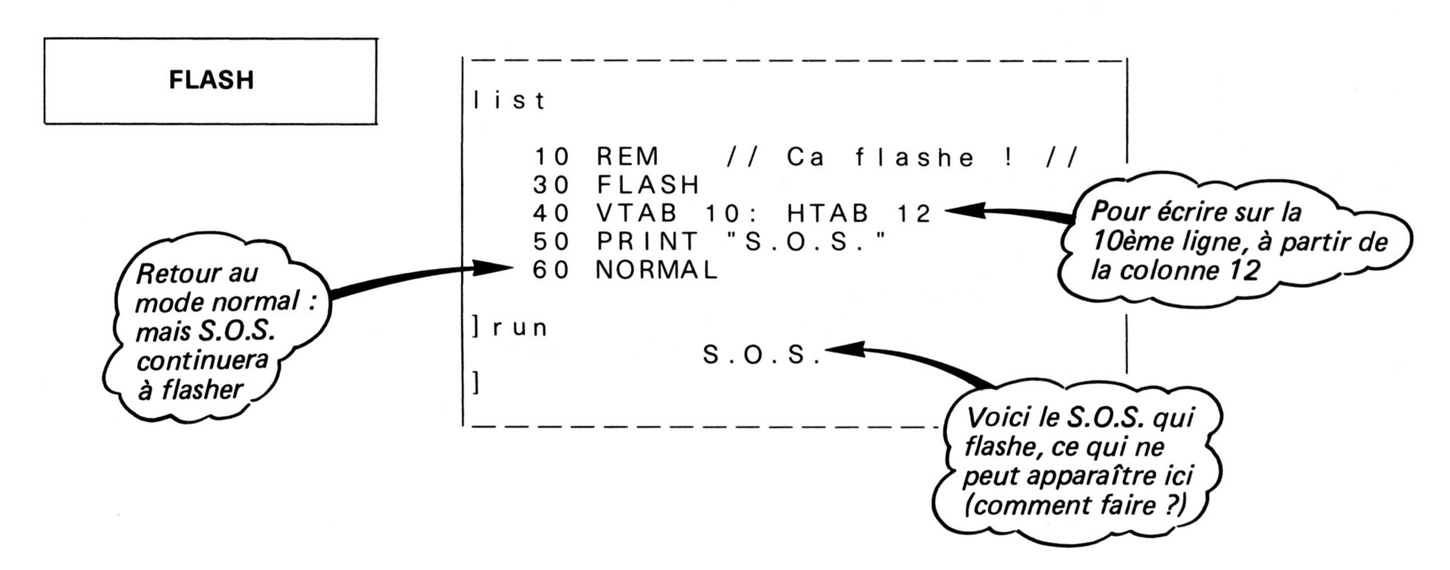
NORMAL



Pour revenir à l'état « normal », frappez simplement :

```
] NORMAL
```

Tout reviendra dans l'ordre, en blanc sur fond noir. Allons plus loin : on va passer automatiquement du normal à l'inverse, et de l'inverse au normal, etc., c'est-à-dire faire clignoter un affichage avec l'ordre FLASH. Entrez ce programme en haut de l'écran et lancez-le, après un CONTROLE/L et un LIST, par exemple :



Voici donc une façon très efficace d'attirer l'attention sur certains messages.

Réponses aux questions sur le chapitre VIII

1. CATALOG affiche les noms des fichiers d'une cassette.

2. L'ordre de sauvegarde est SAVE. Ne confondez pas !

Questions sur le chapitre IX

Les ordres suivants (ou les séquences suivantes) comportent-ils des erreurs, selon vous ?

1. 10 READ a,b,c
   20 PRINT a,b,c
   30 END
   40 DATA 1,2,3

2. Cette séquence fait suite à la précédente :
   50 RESTORE
   60 READ m,n,p
   70 PRINT m,n,p

Et que répondriez-vous à ceci ?

3. En graphique haute définition, la définition est de :
   ☐ 40 par 40 points
   ☐ 160 par 256 points

4. Pour placer de la couleur en haute définition, on commande :
   ☐ PLOT
   ☐ HPLOT

5. Peut-on tracer des droites avec un ordre spécifique, HPLOT, en HGR2 ?
   ☐ Oui
   ☐ Non

6. Peut-on encore accroître l'écran graphique et aller au-delà de 160 par 256 points ?
   ☐ Oui
   ☐ Non

7. Peut-on écrire une donnée directement dans une cellule mémoire précise ?
   ☐ Oui
   ☐ Non

8. Pour agrandir un dessin en haute définition, on commande :
   ☐ ZOOM
   ☐ SCALE

9. Pour faire pivoter une forme, on écrit :
   ☐ MOVE
   ☐ ROT

Réponses page 110

CHAPITRE X

TABLEAUX

Si vous avez beaucoup d'informations à traiter (un recueil d'adresses, des pièces en stocks, des références...), il est très intéressant de les organiser en « tableaux », voici comment. Nous profiterons de ce dernier chapitre de programmation avancée pour vous montrer quelques autres ordres et fonctions, jusqu'à la façon d'avoir l'heure.

Principaux thèmes étudiés

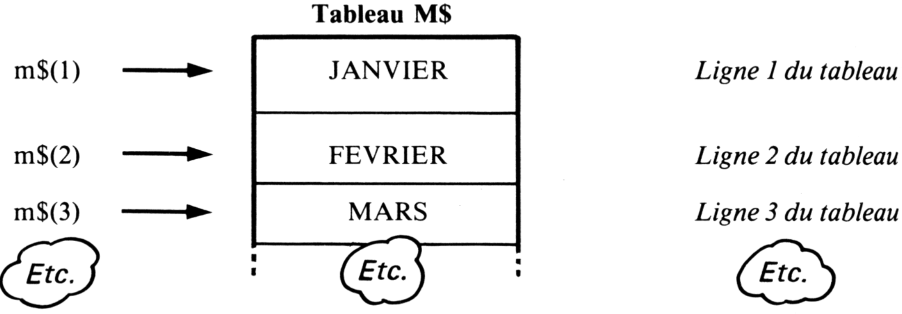
Tableau	ATN
Variable tableau	TAN
Initialisation d'une variable	EXP
Variable indicée	LOG
Variable souscrite	Définition d'une fonction
Dimensionnement	FN
DIM	DEF FN
CLEAR	Variables entières (%)
Tableau mono-dimension	LEN
Tableau multi-dimensionné	LEFT$
Tableau bi-dimensionné	RIGHT$
ABS	MID$
SGN	VAL
SIN	STR$
COS	

1. Le principe des tableaux

Si vous devez travailler avec un nombre limité de variables, numériques ou chaînes, vous leur trouverez facilement des noms et vous pourrez aussi vous en souvenir (c'est très important !). Mais si le nombre de variables devait dépasser la centaine, par exemple, les choses seraient bien plus compliquées. Dans un tel cas, la bonne formule consiste à décerner à chaque groupe de variables homogènes un nom commun, puis à leur attribuer un rang. On va prendre un exemple très simple : les trois mois janvier, février, mars ; on pourrait écrire que A$ = « Janvier », B$ = « Février » et C$ = « Mars », mais ça se compliquerait avec douze mois. Restons-en cependant à trois pour notre exemple.

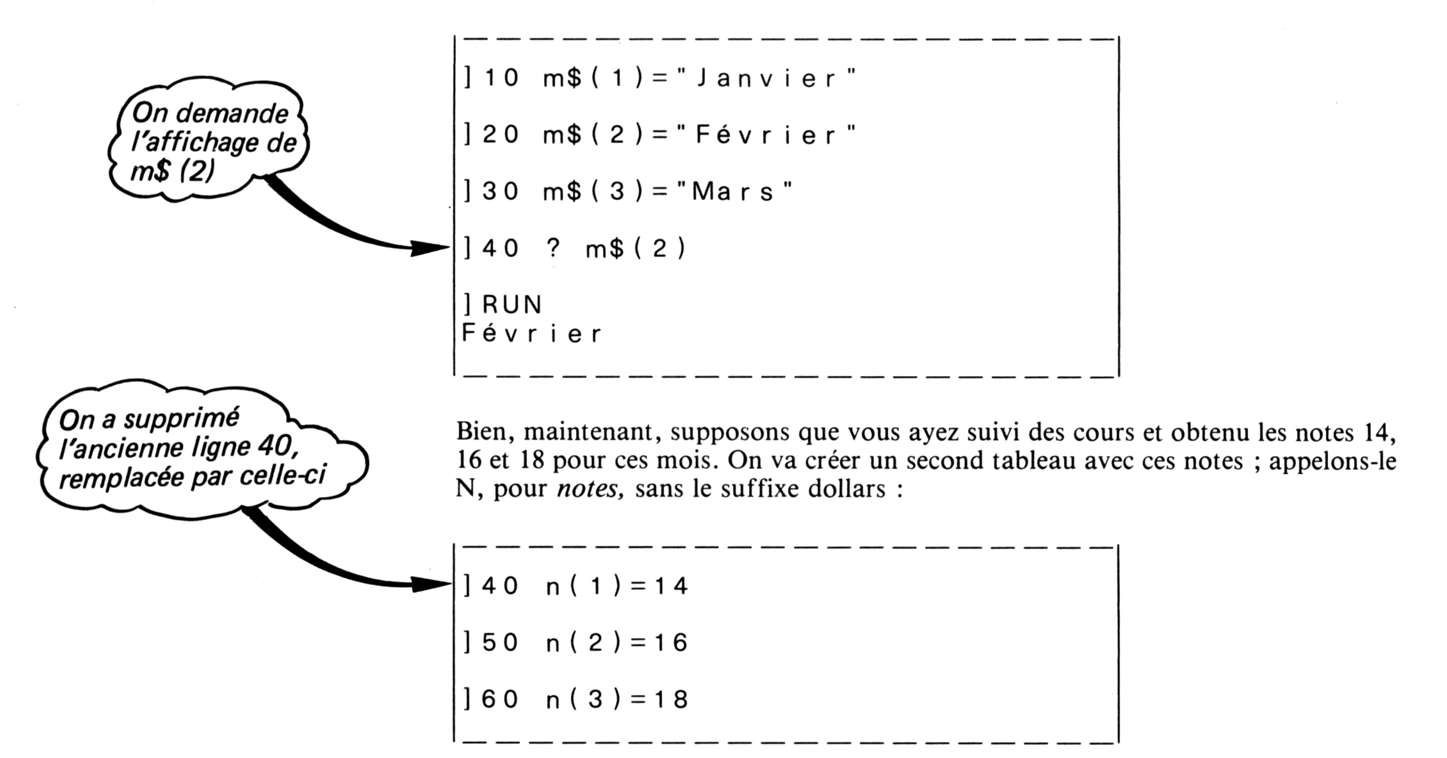
TABLEAU

On appellera globalement M$ ces *m*ois (rappelez-vous que le symbole *dollars* précise qu'il s'agit d'une chaîne de caractères). Mais, de plus, on va leur décerner un numéro d'ordre. Ainsi, *m*$(1) sera Janvier, *m*$(2) sera Février et *m*$(3) sera Mars. C'est déjà bien plus simple. On peut d'ailleurs représenter cela sous forme d'un petit tableau :



VARIABLE TABLEAU

Ainsi, on peut placer dans les cellules de ce tableau (qui sont, en réalité, les cellules de la mémoire de votre ordinateur) des informations quelconques, appelées par ce nouveau type de nom de variable, une *variable tableau*. Elle peut être une chaîne de caractères (et comportera donc le symbole *dollars*) ou numérique (sans le symbole *dollars*). Expérimentons-le :




```
]10 m$(1)="Janvier"

]20 m$(2)="Février"

]30 m$(3)="Mars"

]40 ? m$(2)

]RUN
Février
```

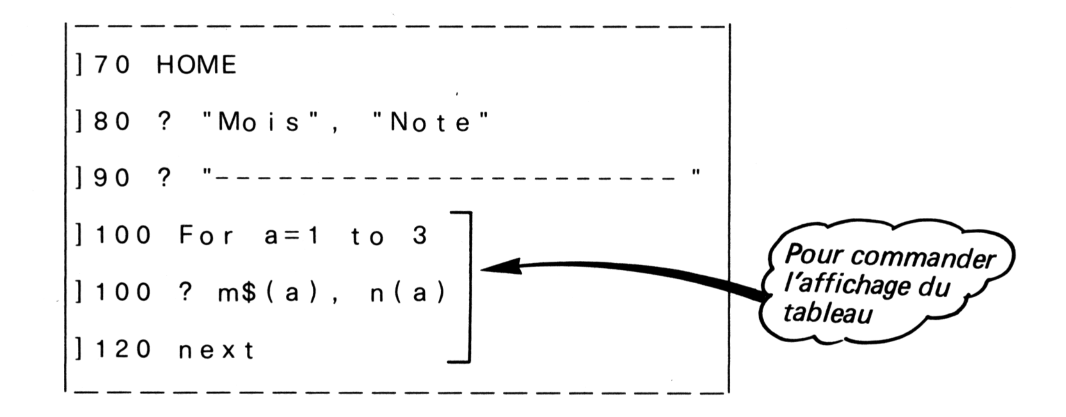
Bien, maintenant, supposons que vous ayez suivi des cours et obtenu les notes 14, 16 et 18 pour ces mois. On va créer un second tableau avec ces notes ; appelons-le N, pour *notes,* sans le suffixe dollars :

```
]40 n(1)=14

]50 n(2)=16

]60 n(3)=18
```

Voulez-vous faire apparaître tout ceci sur l'écran ? Ajoutez ces ordres :




```
]70 HOME

]80 ? "Mois", "Note"

]90 ? "---------------------- "

]100 For a=1 to 3

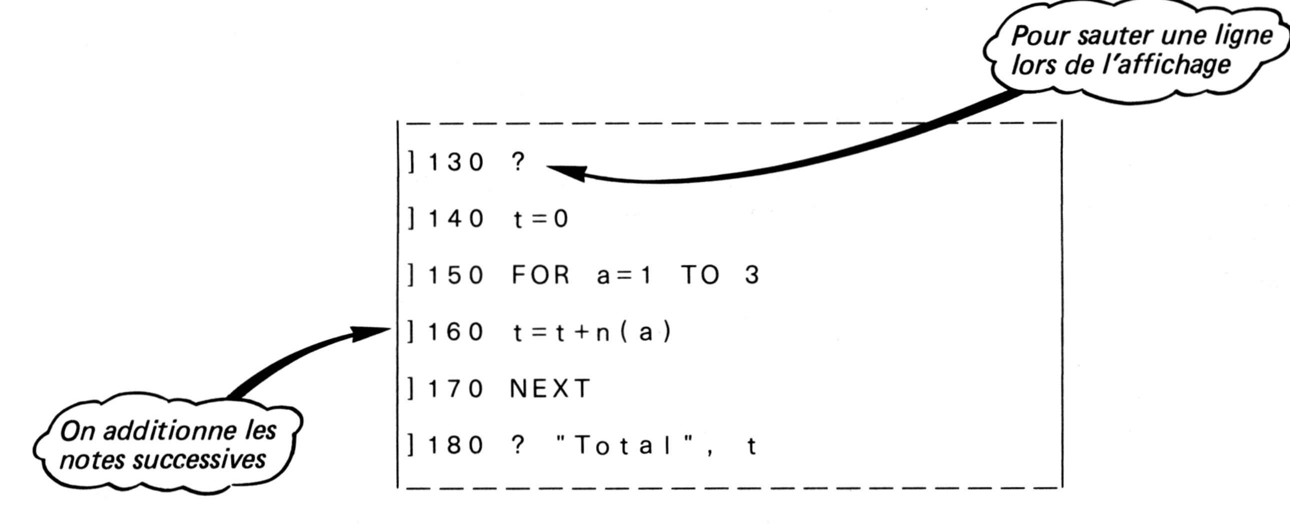
]100 ? m$(a), n(a)

]120 next
```

Si vous faites un RUN, vous allez obtenir un tableau dans lequel les mois et les notes vont être affichés (grâce à la boucle FOR TO) en colonnes en raison de la virgule de tabulation :

```
Mois                Note
------------------------
Janvier             14
Février             16
Mars                18
```

Vous pouvez aussi totaliser les notes. Le total va être appelé *t* ; au départ, il vaut 0, cela va sans dire mais c'est mieux en le disant. Puis, on va ajouter *n*(1) à *t*, puis *n*(2) et enfin *n*(3) en passant une boucle :




```
]130 ?
]140 t=0
]150 FOR a=1 TO 3
]160 t=t+n(a)
]170 NEXT
]180 ? "Total", t
```

Remarquez bien, à nouveau, l'usage du symbole *égal :* il désigne une *affectation* et non une égalité : la *nouvelle* valeur de *t* (à gauche du =) est *l'ancienne* plus la valeur de n(a).

INITIALISATION D'UNE VARIABLE

A la ligne 50, on a d'abord remis *t* à zéro, avant toute addition. C'est ce qu'on appelle « initialiser une variable ».

Voulez-vous, maintenant, établir une moyenne ? Il suffira de diviser le total par le nombre de mois et d'afficher le résultat. On va appeler *r* (pour résultat) cette moyenne :

```
]190 r=t/3
]200 ?
]210 ? "Moyenne", r
]220 ? "Elève doué, mais peut f
aire mieux !"
```

Ce qui donnera, avec un RUN :

```
Mo i s                    No t e
------------------------
J a n v i e r             1 4
F é v r i e r             1 6
Ma r s                    1 8

T o t a l                 4 8

Mo y e n n e              1 6
E l è v e  d o u é ,  ma i s  p e u t  f a i r e  mi e
u x  !
```

La manipulation des tableaux vous paraît-elle claire ? Sachez encore que l'indication de la ligne du tableau, mise entre parenthèses, a le même sens que la notation indicée qu'on utilise dans les textes courants ; dans ceux-ci, on trouverait par exemple n_2 au lieu de *n*(2). Or, l'écriture indicée n'est pas pratique, en informatique, aussi préfère-t-on lui substituer cette écriture avec des parenthèses.

VARIABLE INDICEE

VARIABLE SOUSCRITE

Telle est cependant la raison pour laquelle N est appelée parfois une « variable indicée ».

Le mot *indicé* se dit, en anglais « *subscript* », aussi une mauvaise traduction veut-elle qu'on parle aussi de *variable souscrite.*

2. Le dimensionnement

On vient aussi de créer deux tableaux, l'un de chaînes de caractères (*m*$) et l'autre, numérique *(n).* Leurs constantes pouvaient être introduites via des lignes de DATA, ce qui aurait été plus élégant. On a indiqué la *première* ligne de ces tableaux avec un (1) entre parenthèses : *m*$(1) ou *n*(1) ; or, en pratique, la première ligne commence en zéro. Donc, on aurait pu partir de *m*$(0) et *n*(0) : en pratique, on néglige souvent le rang d'ordre 0 par souci de logique.
Mais jusqu'où peut-on aller ? Le Basic accepte *directement* un tableau allant jusqu'à la ligne 10 ; puisque la première ligne est 0, on dispose d'un total de *onze lignes*. Que faire s'il vous en faut davantage, 12 pour les mois, 365 pour les jours (années non bissextiles) ou 23456 pour les pièces en stock ou les références ? Une seule chose : en informer le Basic en commençant par lui *déclarer* une certaine longueur de tableau. Dans ce cas, le Basic prendra les mesures qui s'imposent et réservera dans la mémoire un nombre suffisant de cellules.

DIMENSIONNEMENT

DIM

Cette réservation est effectuée sur un ordre de « dimensionnement » du tableau, appelé DIM. Elle doit obligatoirement précéder tout travail sur le tableau. Elle ne peut plus être modifiée ensuite.

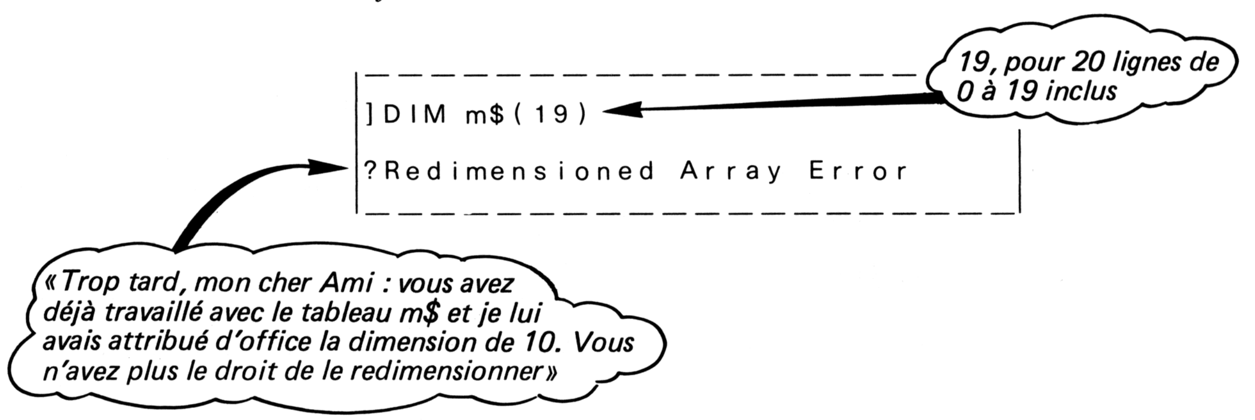
Faites l'essai, en mode direct : dites à l'ordinateur que vous voulez attribuer à *m*$(12) le mot « Décembre » sans avoir réservé de la mémoire avec DIM au préalable :

```
]m$(12)="Decembre"

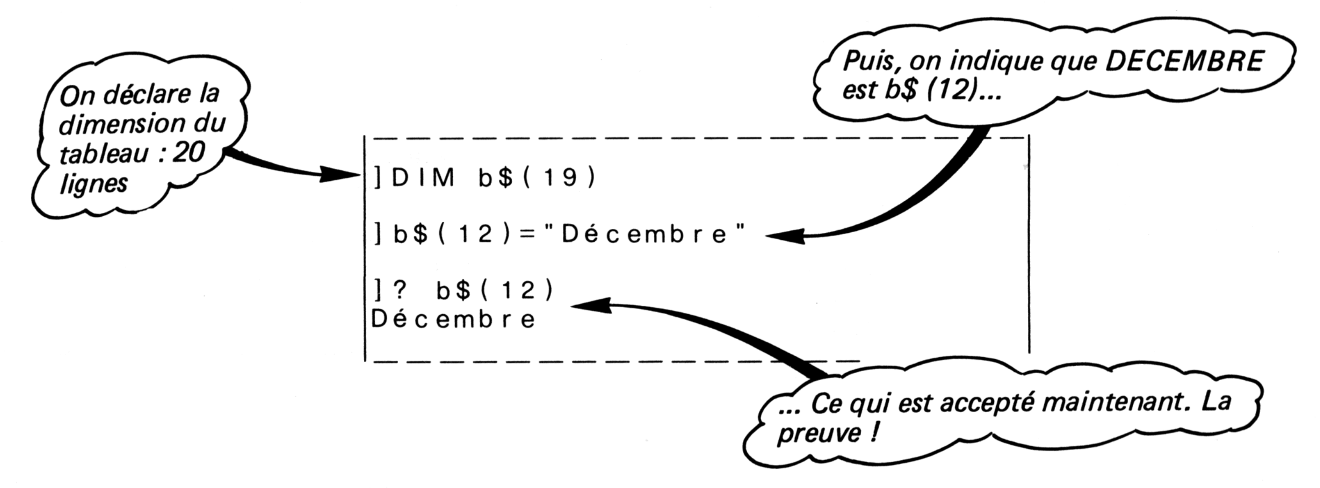
?Bad Subscript Error
```

«Erreur d'indice pour la variable»

Le Basic refuse cette variable. Par contre, si vous « dimensionnez » le tableau au préalable, il l'acceptera. Pour le vérifier, essayez en mode direct de déclarer que *m*$ occupera par exemple 20 lignes : si vous n'avez pas arrêté votre ordinateur entre-temps *et parce que vous avez déjà travaillé* avec le tableau *m*$ auparavant, le Basic *refusera* cette déclaration :



Faisons le même essai avec un autre nom de tableau, par exemple *b*$; ce tableau n'ayant pas encore servi, le Basic va accepter son dimensionnement à 19. Après quoi, on pourra déclarer que *b*$(12) = « Décembre », ce qui donne en mode direct :



CLEAR

Tout ceci trouvera, normalement, place dans un programme. Notez encore qu'il est possible de redimensionner un tableau à la condition d'exécuter au préalable un ordre CLEAR, ce qui aura aussi pour conséquence d'effacer toutes les constantes qui auraient été précédemment créées.

L'ordre CLEAR remet toutes les variables numériques à zéro et « vide » les variables chaînes (elles ne contiennent plus rien ensuite, même pas un zéro).

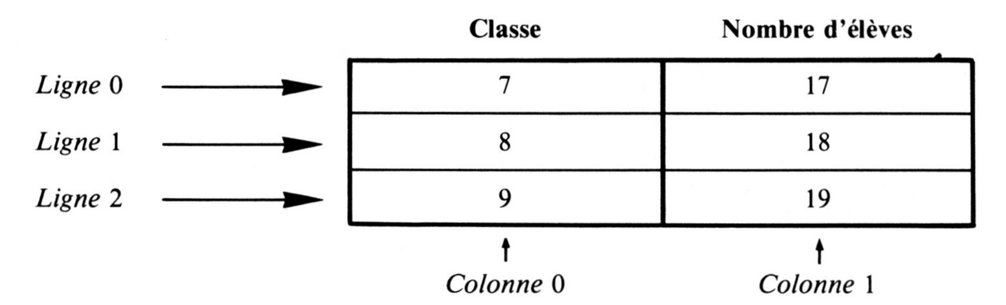
TABLEAU MONO-DIMENSION
MULTI-DIMENSION

3. Tableaux multi-dimensions

Chacun des tableaux qu'on a examinés (*m*$ ou *n*) ne comportait qu'une *seule* colonne de données : c'est d'ailleurs ce qu'on appelle *une liste,* ou encore un tableau *mono-dimension,* ou encore *mono-dimensionné.* Il a donc fallu *deux* tableaux pour créer 2 colonnes. Mais on peut imaginer des tableaux à plusieurs dimensions, par exemple à deux dimensions, comportant *deux* ou davantage de colonnes (comme un damier).

Le besoin s'en fera sentir si l'on fait des statistiques, des comptes, si l'on établit une liste de classes et d'élèves, ou alors pour compter les chambres d'un hôtel par étage, etc. Supposons qu'on compte des élèves : la 7e classe en compte 17, la 8e en compte 18 et la 9e en compte 19. Notre *unique* tableau sera (avec deux colonnes pour *un seul* tableau, cette fois) :

Tableau a(classe,élèves)



	Classe	Nombre d'élèves
Ligne 0 →	7	17
Ligne 1 →	8	18
Ligne 2 →	9	19
	↑ *Colonne* 0	↑ *Colonne* 1

Appelons ce tableau *a ;* cette fois, il dispose de *deux* dimensions, pour les lignes et pour les colonnes. Ici, on a numéroté lignes et colonnes à partir de Ø, simplement pour vous montrer que c'était possible. Comment les cellules de ce tableau vont-elles être désignées ?

TABLEAU BI-DIMENSIONNE

Tableau a(c,e)

Ligne 0	a(0,0)	a(0,1)
Ligne 1	a(1,0)	a(1,1)
Ligne 2	a(2,0)	a(2,1)

Remarquez bien la façon d'écrire :

Nom du tableau (ligne, colonne)

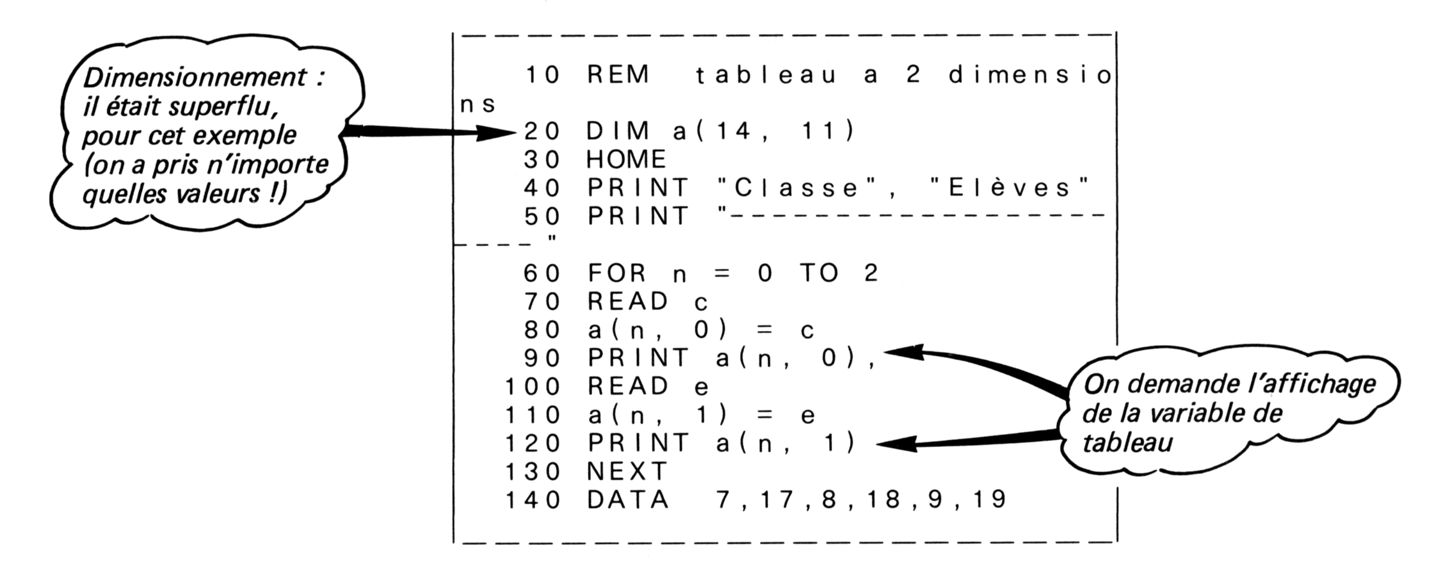
Si l'on négligeait les lignes et colonnes d'ordre 0, on pourrait commencer par *a*(1,1) et non *a*(0,0), ce qui ne changerait rien quant au fond.

Créons un court programme qui introduise les données dans une ligne DATA. Elles sont lues avec READ dans une boucle, affectées aux variables tableaux ; simultanément, on demande l'affichage des variables tableaux (et non des données directement, pour les besoins de la démonstration !). En raison du nombre de lignes et de colonnes inférieur à 11, on n'a pas besoin du DIM mais on l'a ajouté, ici, pour montrer son emploi en supposant qu'on veuille disposer de 15 rangées et de 12 colonnes.

Notez que la déclaration de dimension indique :

- **le nombre de lignes et**
- **le nombre de colonnes dont on veut disposer.**

Remarquez la façon dont, aux lignes 80 et 110 de ce programme, on affecte les constantes aux variables : on lit d'abord *c* (*classe*) et on l'affecte à la colonne 0 de la ligne *n* en cours ; puis, on lit *e* et on l'affecte à la colonne 1 de la ligne *n* en cours :




```
   10 REM  tableau a 2 dimensio
ns
   20 DIM a(14, 11)
   30 HOME
   40 PRINT "Classe", "Elèves"
   50 PRINT "------------------
---- "
   60 FOR n = 0 TO 2
   70 READ c
   80 a(n, 0) = c
   90 PRINT a(n, 0),
  100 READ e
  110 a(n, 1) = e
  120 PRINT a(n, 1)
  130 NEXT
  140 DATA  7,17,8,18,9,19
```

Ne vous embrouillez pas avec les diverses variables et remarquez que, cette fois, on n'a fait appel qu'à un *unique* tableau pour afficher sur deux colonnes, ce qui est possible puisque les variables sont homogènes (*toutes numériques*). Un RUN donnera :

```
Classe               Elèves
----------------------------
7                    17
8                    18
9                    19
```

Comme précédemment, on peut faire le total des élèves ; il suffit d'ajouter :

On n'ajoute que les valeurs continues dans la colonne de rang 1

```
]150  t=0

]160  FOR n=0  TO 2

]170  t=t+a(n,1)

]180  NEXT

]190  ?:? "Le  total  des  élèves  e
st:  ";t
```

Un RUN donnerait :

```
Classe               Elèves
----------------------------
7                    17
8                    18
9                    19

Le total des élèves est:  54
```

Si le tableau à deux dimensions comportait 4 colonnes, voici comment ses cellules seraient repérées (pour deux rangées ici) :

Tableau f(1,3)

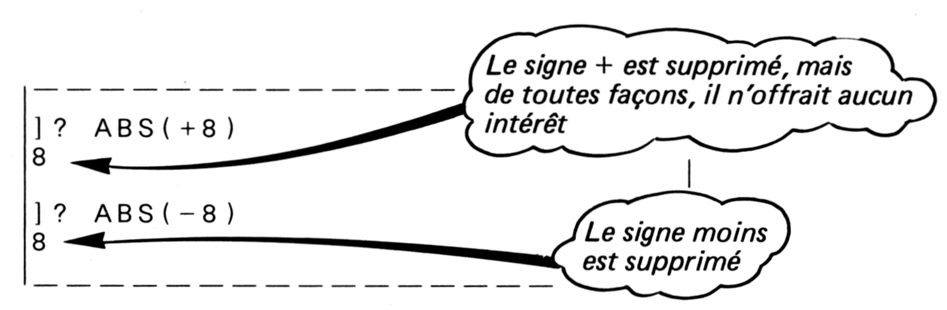
f(0,0)	f(0,1)	f(0,2)	f(0,3)
f(1,0)	f(1,1)	f(1,2)	f(1,3)

On pourrait, de la même façon, concevoir un tableau à 3 dimensions, dimensionné avec DIM F(*hauteur,largeur,profondeur*) qu'on pourrait représenter par un cube, ou encore des tableaux à 4 ou 5 dimensions ou davantage (des « hypercubes »), mais gageons qu'ils ne sont pas d'un besoin général quotidien.

4. Fonctions et définition de fonctions

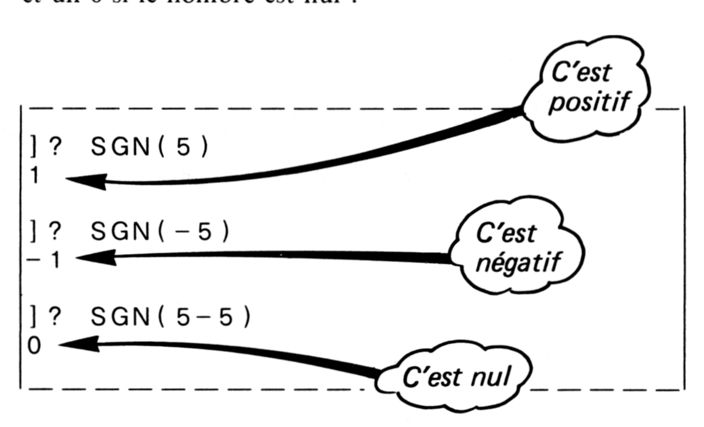
ABS

Puisque nous évoquons les valeurs numériques, nous pouvons vous donner une nouvelle liste de fonctions du Basic typiquement arithmétiques. Ainsi, ABS fournit la valeur *absolue* d'un nombre, c'est-à-dire qu'elle le débarrasse de son signe, *plus* ou *moins*. Essayez, en mettant les nombres entre parenthèses :



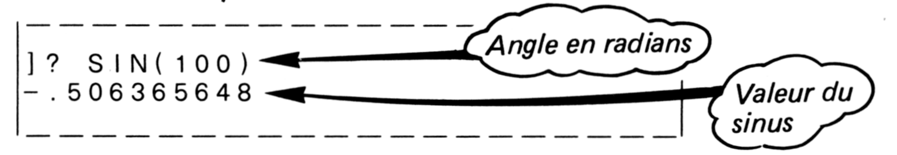
SGN

L'ordre *signe* (SGN), lui, renvoie un 1 si le signe est positif, un −1 s'il est négatif, et un 0 si le nombre est nul :



SIN
COS
ATN
TAN
EXP
LOG

Par ailleurs, vous disposez de toute une série de fonctions logarithmiques, par exemple SIN qui renvoie le sinus d'un angle donné en *radians* (et non en degrés !) :



Nous vous ferons grâce des autres fonctions telles que COS pour le cosinus, ATN pour l'arc tangente, TAN pour tangente... qui passionneront certainement les amateurs de trigonométrie, ainsi que des exponentielles (EXP), logarithmes LOG, etc., en sachant pertinemment que les amateurs de mathématiques se sentiront frustrés.

Un ordre, toutefois, va retenir encore un instant notre attention ; c'est celui qui permet de *définir une fonction mathématique* complexe. Prenons-en une simple : la surface d'un cercle qui vaut, chacun le sait πR^2 ce qui s'écrira en Basic 3.14 *R^2. Rappelez-vous aussi que la tête de flèche verticale sert à élever un nombre à une puissance.

De ce fait, la fonction (notée FN) calculant la surface *s* du cercle dépend de la variable *r*, ce qu'on écrira FN *s(r)*. On définira (on posera) cette fonction une fois pour toute en écrivant DEF FN s(r) = 3.14 *r^2 puis on demandera à l'opérateur d'introduire une valeur pour *r* et on commancera à la fois l'exécution du calcul et l'affichage du résultat :

DEFINITION D'UNE FONCTION
DEF FN
FN

```
   10 REM  definition d'une fon
ction
   20 DEF FN s(r) = 3.14*r^2
   30 INPUT "Le rayon vaut:"; r

   40 PRINT "La surface du cerc
le est:"; FN s(r)
   50 GOTO 20

]run
Le rayon vaut:5
La surface du cercle est:78.5
Le rayon vaut:

?Break In 30
```

Après un CONTROLE/C

Cet ordre peut vous simplifier grandement les calculs.

5. Variables entières

VARIABLES ENTIERES
%

Parfois aussi, vous pourrez simplifier des calculs en passant par des *variables entières*. Supposons qu'une variable numérique s'appelle *a* ; si l'on veut imposer qu'il s'agisse de constantes entières (des nombres entiers, sans partie décimale), on signifiera que la variable est une variable *entière* avec le suffixe % ; ainsi, on écrira *a*%. Pour comprendre ce que cela donne, essayez (après avoir fait un CLEAR par sécurité, pour effacer les variables antérieures que vous avez créées : il ne faut pas mélanger les torchons et les serviettes) :

```
]CLEAR

]a%=5

]? a%
5

]? a
0

]b%=4.3

]? b%
4
```

On attribue un nombre entier à une variable entière. Pas de problème

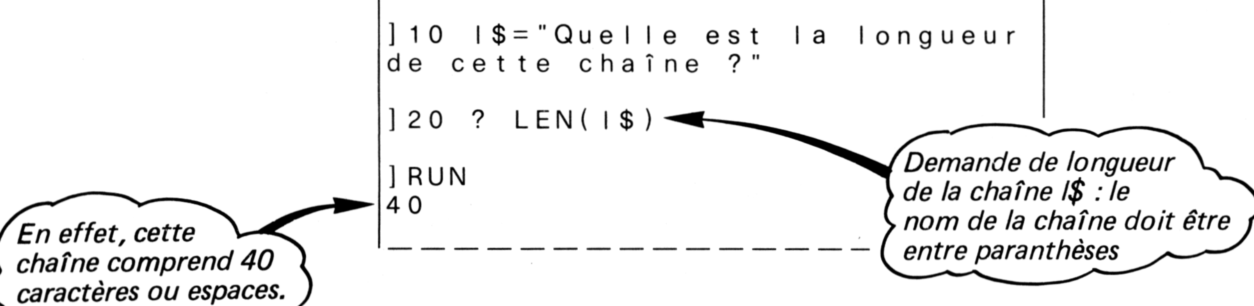
Pour l'ordinateur, a % et a tout court sont deux noms différents de variable ; a n'ayant reçu aucune affectation vaut 0

Voyez ce qui se passe avec b % : cette variable ne retient que la partie entière du nombre proposé.

6. Pour jongler avec les chaînes

LEN

Arrêtons-nous encore un instant sur les chaînes, puisque des instructions spécifiques ont été prévues pour les manipuler. Ainsi, vous voulez savoir combien de caractères comporte la chaîne : « QUELLE EST LA LONGUEUR DE CETTE CHAINE » ; ne les comptez pas, demandez-le au Basic avec l'ordre LEN qui vient de « *length* », longueur. N'oubliez pas un NEW au préalable :

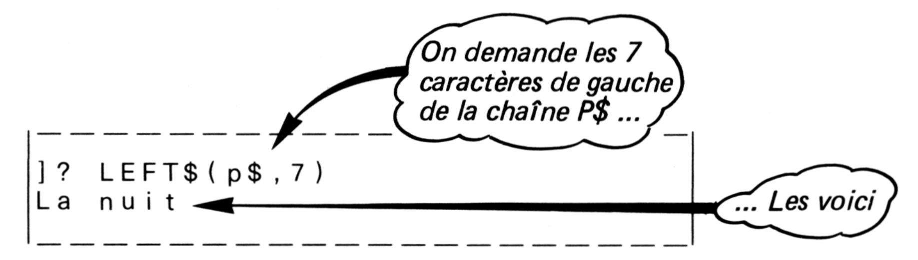


Déclarons maintenant, et en mode direct, le proverbe P$ qui sera « La nuit, tous les chats sont gris » :

```
]p$="La nuit, tous les chats so
nt gris"
```

LEFT$

On veut extraire de cette chaîne les deux seuls mots de gauche, par conséquent les sept premiers caractères de gauche. Sachant que *gauche* se dit « *left* », en anglais, on codera un ordre LEFT$ (avec le suffixe *dollars*) suivi d'une virgule puis du nombre de caractères désirés, et l'on fera (en mode direct) :



RIGHT$

Voulez-vous les 19 caractères de droite ? Sachant que *droite* se dit « *right* », en anglais, vous comprendrez cet ordre :

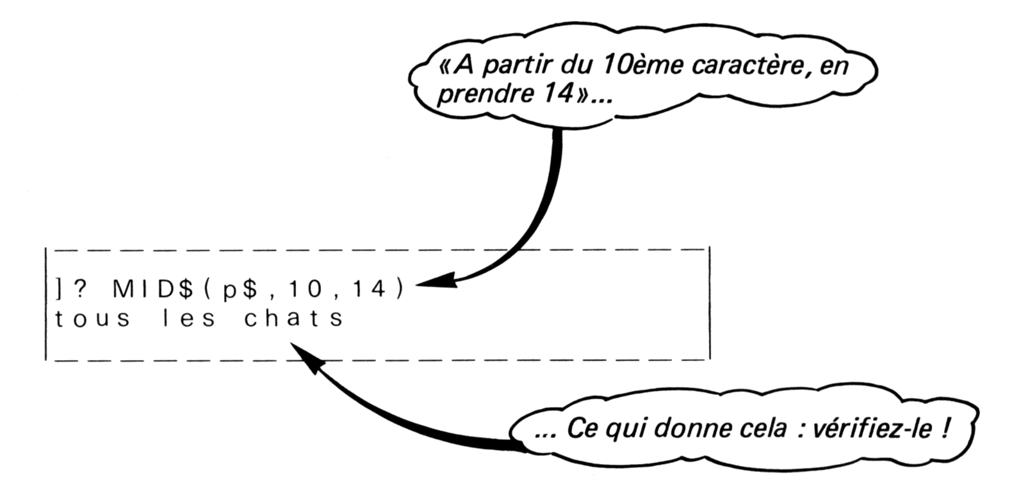
```
]? RIGHTS(p$,19)
les chats sont gris
```

Remarquez bien la syntaxe de ces ordres :

ordre LEFT$ ou RIGHT$ (nom de la chaîne, longueur désirée)

On peut aussi viser au centre en indiquant, cette fois, à partir de quel caractère on veut la tranche de phrase :

MID$



De tels ordres de manipulation de chaînes peuvent procurer facilement des résultats bizarres. Voici un exercice de style qui l'illustre ; examinez ce programme et devinez ce qu'il doit fournir :

```
]5 REM incantation

]10 HOME

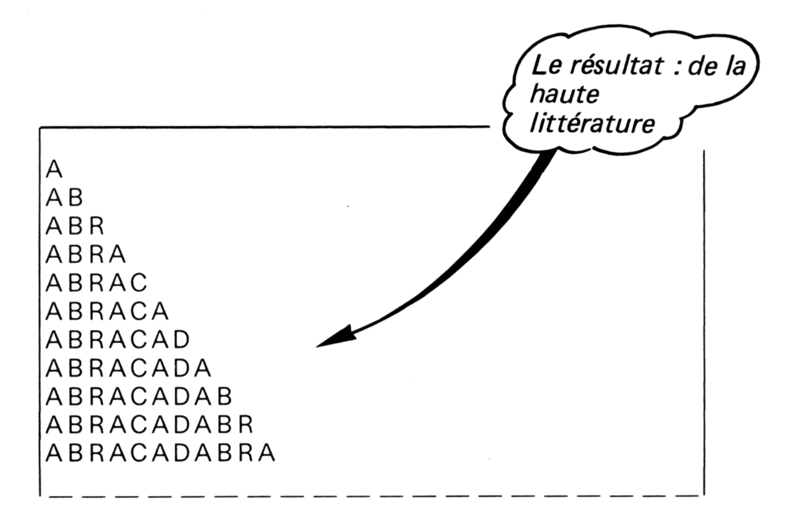
]20 a$="ABRACADABRA"

]30 FOR n=1 TO 11

]40 ? LEFT$(a$,n)

]50 NEXT
```

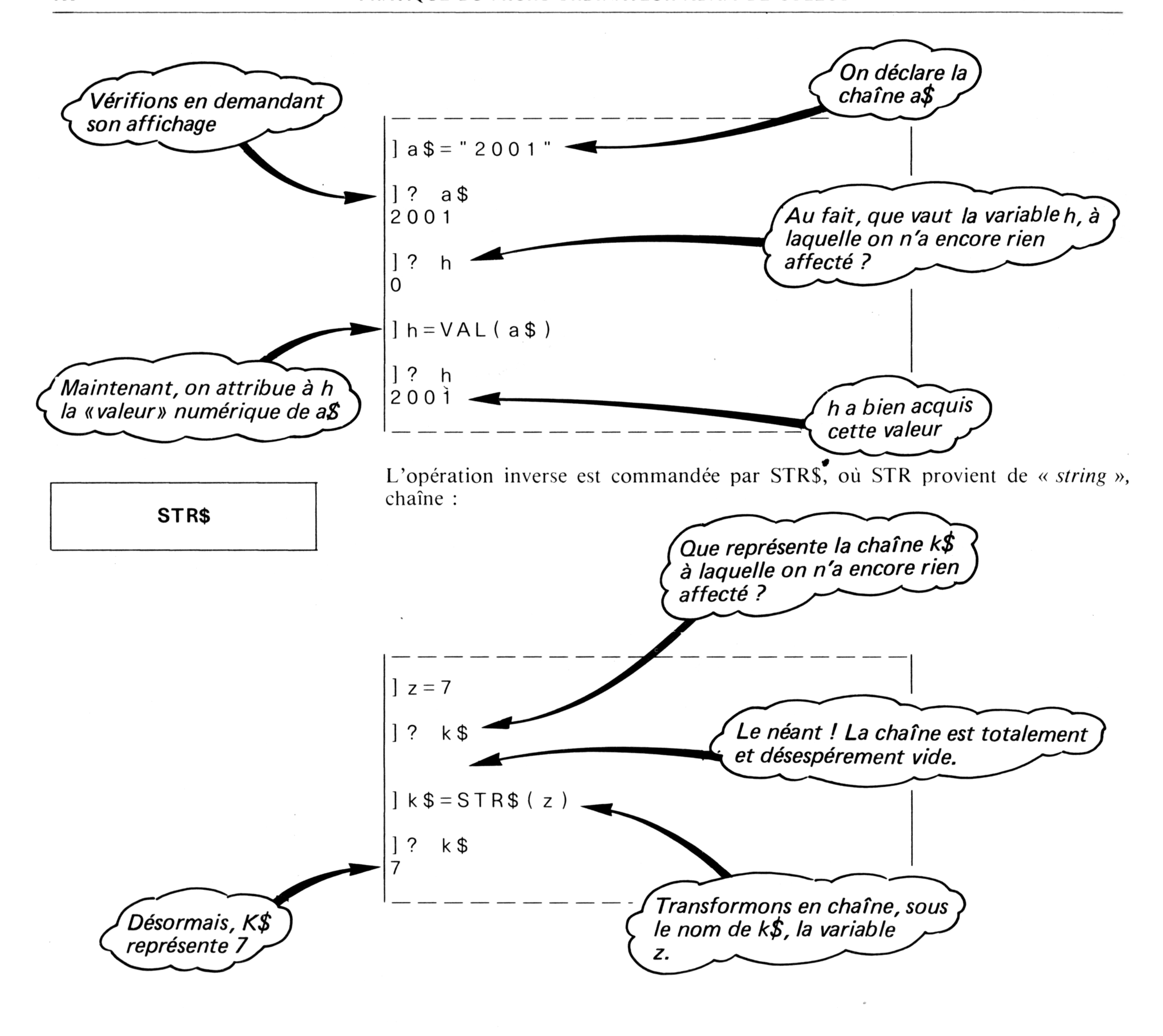
Si vous n'avez pas deviné ce qu'un RUN donne, voici :



Essayez par vous-même de créer un programme qui affiche en marche arrière, ou alors qui parte du centre d'une phrase et s'étale symétriquement de part et d'autre, ou qui utilise des symboles graphiques pour cela.

VAL

Ajoutons encore à cette série deux ordres. Si une chaîne de caractères ne contient qu'une valeur numérique, vous pouvez lui restituer son caractère numérique (donc, vous dispenser du symbole *dollars* et des guillemets) en ne conservant que cette valeur numérique (d'où l'ordre VAL) ; voici la séquence à expérimenter en mode direct :



STR$

L'opération inverse est commandée par STR$, où STR provient de « *string* », chaîne :

Notez qu'on aurait pu travailler avec des noms de variables tels que Z pour la numérique et Z$ pour la chaîne, puisque le Basic considère qu'il s'agit de noms différents, mais c'eût été moins clair.

Remarquez également qu'une variable chaîne à laquelle on n'a pas affecté de constante est vide, ne contient pas même un zéro : c'est le néant. Tout comme un LIST demandé alors qu'il n'y a pas d'instructions en mémoire.

Ajoutons que l'usage de ces deux dernières instructions n'est pas des plus fréquents.

Questions sur le chapitre X

C'est le jeux des erreurs. Trouvez celle qui existe, s'il y en a une, dans ces lignes, puis répondez aux autres questions :

1. m$(567) = 32

2. DIM t(10,10,10)

3. 10 t(27) = 333
   20 DIM t(50)

4. a(5,7) correspond à une cellule sur :
   ☐ La rangée 5, colonne 7
   ☐ La colonne 5, rangée 7

5. ? ABS(− 8) donnera
   ☐ 8
   ☐ − 8

6. A-t-on le droit de définir la fonction suivante :
   FNX(t) = 3.14^t + 127 * (7 − t)

7. Si l'on attribue 3.14 à la variable P% puis que l'on demande l'affichage de P%, on obtiendra :
   ☐ 3.1416
   ☐ 3

8. Que donnera LEN(BONJOUR) ?
   ☐ 7
   ☐ Une erreur

9. Que donnera :
   10 a$ = " BONJOUR "
   20 PRINT LEFT$(a$,3)
   ☐ BON
   ☐ JOUR

Réponses page 162

Réponses aux questions sur le chapitre X

1. La variable tableau m$ peut très bien s'adresser à la ligne 567, mais il faut lui affecter une constante chaîne, et non une constante numérique. Si vous teniez absolument à la constante 32, il fallait écrire

m$(567) = « 32 »

2. Ici, on a dimensionné un tableau à 3 dimensions (un cube) pour des dimensions de 10 par arête. C'est tout à fait inutile : le Basic le fait d'office, à défaut d'avis contraire. Mais il n'y aura pas d'erreur.

3. Non : le dimensionnement doit *précéder* tout travail avec le tableau. Dès que vous évoquez un tableau pour la première fois (ligne 10), le Basic le dimensionne d'office à 10 et ne veut plus entendre parler de dimensionnement.

4. a(5,7) correspond à une cellule sur la rangée 5, colonne 7.

5. La valeur absolue de −8 est 8 (et non −8).

6. Oui, c'est absolument correct. Maintenant, qu'allez-vous faire de cette fonction bizarre ?

7. P% caractérise une variable entière ; par conséquent, la partie décimale disparaît et l'affichage de P% donne ici 3.

8. Oui, une erreur car on n'a pas mis la chaîne entre guillemets et de surcroît, on n'a pas dit PRINT.

9. Vous obtiendrez BON (les 3 caractères de gauche de la chaîne a$).

ET MAINTENANT

Vous voici arrivé au terme de ce livre. Nous vous en félicitons. Vous avez maintenant acquis l'essentiel de ce qu'il faut savoir pour exploiter votre Adam : le traitement de texte, l'utilisation des cassettes digitales, celle des modules préprogrammés (de jeux, entre autres), la programmation en Basic.

Sachez toutefois que nous n'avons pu tout vous dire. Notre style d'explication aurait obligé à doubler le volume de ce livre, et par conséquent son prix aussi, ce que vous ne souhaitez peut-être pas. Ainsi nous n'avons pas développé le rôle de certaines instructions secondaires, ni celles d'aide au développement de programmes, non plus que toutes celles relatives aux fichiers. D'autre part, nous n'avons pas abordé la programmation des sons. En fait, nous avons éliminé une partie des instructions qui s'adressent à l'expert : que vous deviendrez peut-être.
Dans ce cas, reportez-vous aux manuels qui vous sont livrés avec Adam et qui vous apporteront les informations supplémentaires, sous une forme succincte sinon lapidaire, manquant le plus souvent d'exemples développés. Mais avec ce que vous savez déjà, vous imaginerez le complément.

Nous espérons qu'au cours de cette lecture, vous nous avez quittés ici ou là pour partir sur vos idées propres, créer ou jouer. Continuez ainsi : c'est la pratique qui vous fait encore défaut. Sachez aussi qu'un nouveau programme qui vient d'être inventé comporte immanquablement des erreurs (à moins que le programmeur ne soit un surdoué, ou un Martien) ; aussi ne vous inquiétez pas de celles que vous commettrez : apprenez à les corriger.

Mais au fait, disposez-vous déjà du micro-ordinateur Adam ? Si ce n'est encore fait, vous pouvez réparer cet oubli. Il ne s'agit là que d'un modeste problème de trésorerie pour lequel l'auteur ne peut vous aider. L'outil informatique sera ainsi démystifié et participera à votre promotion et à votre efficacité. Ce sera lui qui vous permettra :

- *De faire du traitement de texte, d'envoyer les circulaires de votre club, de faire le rappel de vos factures impayées...*

- *De rédiger des devoirs, des rapports, des études, des notes de services...*

- *D'acquérir la culture informatique, qui est celle de l'« l'honnête homme du XXIe siècle »...*

- *D'apprendre la programmation, le Basic, ainsi que l'art d'un raisonnement rigoureux et de type nouveau.*

- *D'étudier toutes sortes de choses, à partir de programmes tout faits (et non seulement l'arithmétique, l'histoire, la grammaire ou la géographie).*

- *Ah oui, nous allions oublier : de jouer, aussi. Mais cela, vous y auriez pensé de vous-même.*

- *Cette liste n'est nullement limitative. Vous la compléterez et inventerez bien d'autres applications. Nous faisons confiance à votre génie : faites alors confiance à Adam, qui met effectivement à la portée de tous un système dont les applications semblent ne pas connaître de limites.*

MESSAGES D'ERREUR

Vous en recevrez tout votre saoûl ! Voici leur signification à tous (mais certains s'adressent aux experts).

I. SUR LES FICHIERS	
Message	**Signification**
Language not available	Langáge non disponible. Restez donc avec le Basic et sa cassette !
Range error	La gamme permise est excédée
Write protected	Vous ne pouvez écrire en cet endroit
End of data	Le fichier de texte est terminé
Volume mismatch	Vous avez fourni un paramètre incorrect de volume
File not found	Ce fichier n'existe pas
I/O Error	Erreur d'entrée ou de sortie
Disk full	Votre cassette est pleine à déborder
File locked	Fichier verrouillé : vous ne pouvez que le lire
Syntax error	Erreur de syntaxe
No buffers available	Plus de buffers
File type mismatch	Erreur de commande avec fichiers numériques
Program too large	Votre mémoire centrale déborde
Not direct command	Ne peut être utilisé en mode direct. Passez au mode programme
Control buffer overflow	Votre buffer déborde

II. SUR LES PROGRAMMES	
Message	**Signification**
Next without for	Un NEXT mais pas de FOR
Syntax error	Erreur de syntaxe
Return without Gosub	Un RETURN sans son GOSUB
Out of data	Plus de données à lire avec un READ
Illegal quantity	Valeur bizarre et interdite
Overflow	Un résultat astronomique ou infinitésimal
Out of memory	La mémoire déborde
Stack overflow	La pile déborde
Undef'd statement	Cette ligne n'existe pas
Bad subscript	Ne se trouve pas dans le tableau
Redim'd array	Combien de fois faut-il vous le redire ? On ne peut pas redimensionner un tableau.
Divide by zero	Interdiction de diviser par zéro
Type mismatch	Numérique ou chaîne de caractères ? Il faudrait accorder les violons
String too long	Une chaîne de plus de 255 caractères
Formula too complex	Soyez moins compliqué
Undef'd fonction	Vous avez omis de définir cette fonction
Illegal function assignement	Mais c'est un mot réservé, interdit, ça !
Illegal mode	Interdit aussi, mais en mode direct
Break	Après action sur CONTROLE/C
Can't continue	La machine vous le dit : elle ne peut pas continuer ainsi.

FONCTIONS D'ÉDITION AVEC LA TOUCHE CONTROLE

Maintenez la touche CONTROLE enfoncée et :

Frappez	Vous obtiendrez :
C	L'arrêt d'un affichage qui défile sur l'écran, ou l'arrêt de l'exécution d'un programme. Vous pourrez poursuivre avec CONT.
L	L'effacement de l'écran.
N	La possibilité d'insérer du texte supplémentaire.
O	L'effacement du caractère marqué par le curseur.
P	Une sortie sur l'imprimante de tout ce qui se trouve sur l'écran (c'est une « recopie » d'écran).
S	Vous « gelez » l'écran sans sortir de votre programme. Pour repartir, refrappez la même combinaison.
X	Vous supprimez une « entrée » erronée : une barre oblique inversée se place là où se trouve le curseur ; celui-ci revient à la ligne. La ligne précédente est ignorée par le Basic.

CODE ASCII

Le code ASCII traduit, en numérique, les caractères disponibles au clavier. ASCII provient de *American Standard Code for Information Interchange.* Ce code respecte une certaine organisation :

• De 0 à 31, en décimal, apparaissent les fonctions de commande, obtenues avec CONTROLE ou ESC.

• Les chiffres de 0 à 9 se traduisent par 48 à 57 en décimal.

• Les lettres vont de 65 (pour A) à 90 (pour Z).

• A partir de 129 figurent des caractères graphiques ou alphanumériques en vidéo inversée.

Décimal	Caractère ou ordre	Frappez CONTROLE et	Décimal	Caractère ou ordre	Décimal	Caractère ou ordre	Décimal	Caractère ou ordre
0	NULL	@	32	SPACE	64	@	96	‘
1	SOH	A	33	!	65	A	97	a
2	STX	B	34	"	66	B	98	b
3	ETX	C	35	#	67	C	99	c
4	ET	D	36	$	68	D	100	d
5	ENQ	E	37	%	69	E	101	e
6	ACK	F	38	&	70	F	102	f
7	BEL	G	39	’	71	G	103	g
8	BS	H ou ←	40	(	72	H	104	h
9	HT	I	41	)	73	I	105	i
10	LF	J	42	*	74	J	106	j
11	VT	K	43	+	75	K	107	k
12	FF	L	44	,	76	L	108	l
13	CR	M ou RETURN	45	—	77	M	109	m
14	SO	N	46	.	78	N	110	n
15	SI	O	47	/	79	O	111	o
16	DLE	P	48	0	80	P	112	p
17	DC1	Q	49	1	81	Q	113	q
18	DC2	R	50	2	82	R	114	r
19	DC3	S	51	3	83	S	115	s
20	DC4	T	52	4	84	T	116	t
21	NAK	U ou →	53	5	85	U	117	u
22	SYN	V	54	6	86	V	118	v
23	ETB	W	55	7	87	W	119	w
24	CAN	X	56	8	88	X	120	x
25	EM	Y	57	9	89	Y	121	y
26	SUB	Z	58	.	90	Z	122	z
27	ESCAPE	[	59	;	91	[	123	{
28	FS	\	60	<	92	\	124	:
29	GS	]	61	=	93	]	125	{
30	RS	^	62	>	94	^	126	/
31	US	_	63	?	95	—	127	DEL

MOTS RÉSERVÉS

Les mots réservés sont des mots dont le Basic se réserve l'usage. On y retrouve donc tous les ordres que nous avons étudiés dans ce livre, avec quelques petites choses de plus. Vous n'avez pas le droit de les utiliser comme nom de variable. Si vous le faites, vous le saurez vite car le Basic vous les refusera. De toutes façons, nous vous avons conseillé d'utiliser pour nom de variable une unique lettre, éventuellement suivie d'un chiffre ou, à la rigueur, d'une seconde lettre. Mais méfiez-vous des mots réservés en deux lettres. En voici la liste presque exhaustive.

ABS
AND
APPEND
ASC
ATN
BLOAD
BRUN
BSAVE
CALL
CATALOG
CHR$
CLEAR
CLOSE
COLOR
CONT
COS
DATA
DEF FN
DEL
DIM
DRAW
END
EXP
FN
FOR
FRE
GET
GOSUB
GOTO
GR
HCOLOR
HGR2
HIMEM
HLIN
HOME
HPLOT
HTAB
IF
INIT
INPUT
INT
INVERSE
LEFT$
LEN
MET
LIST
LOAD
LOCK
LOG
LOMEM
MID$
MON
NEXT
NEW
NOMON
NORMAL
NOT
NOTRACE
ONERR
ON
OPEN
OR
PDL
PEEK
PLOT
POKE
POP
POS
POSITION
PRINT
PR#
RANDOM
READ
RECOVER
REM
RENAME
RESTORE
RESUME
RETURN
RND
RIGHT$
ROT
RUN
SAVE
SCALE
SIN
TAN
THEN
USER
XDRAW

MOTS CLÉS

Tout au long de ce livre, nous avons encadré des mots, expressions ou idées essentiels. Nous vous invitons à vous y reporter si besoin est. En voici la liste avec le numéro de leur page.

TABLE DES MATIÈRES

S.E.C.F.

EDITIONS RADIO

Service lecteurs

(à retourner à S.E.C.F.-Éditions Radio, 9, rue Jacob, 75006 Paris)

Pour nous permettre de vous proposer des ouvrages toujours meilleurs, nous souhaiterions recevoir vos critiques, appréciations et suggestions sur le présent livre :

Quels sont les ouvrages (thème, sujet, niveau) que vous souhaiteriez voir publier par notre société ?

Nous vous remercions de votre confiance et de votre coopération.

S.E.C.F.-Éditions Radio

Je désire recevoir gratuitement et sans engagement (mettre une croix dans la case) :
☐ Votre catalogue général (Electronique professionnelle et grand public, Informatique, Hi-Fi, Vidéo)
☐ Votre catalogue spécial informatique.

Nom : ______________________ Prénom : ______________________
Adresse : ______________________

Secteur d'activité et fonction : ______________________

CENTRES D'INTÉRÊTS

☐ Electronique professionnelle
☐ Electronique de loisirs
☐ Vidéo
☐ Hifi, CB...
☐ Micro-informatique professionnelle
☐ Micro-informatique de loisirs
☐ Autres : ....
....

Voir au dos "Correspondance Auteurs"

S.E.C.F.



EDITIONS RADIO

Service lecteurs

Correspondance auteurs

(à retourner à S.E.C.F.-Éditions Radio, 9, rue Jacob, 75006 Paris)

Pour toute demande d'éclaircissements techniques relatifs à ce livre, formulez ci-dessous vos questions, avec le maximum de précisions. Nous les transmettrons à l'auteur qui ne manquera pas de vous répondre directement :

Nom : ______________________ Prénom : ______________________

Adresse : ______________________

PRATIQUE
du micro-ordinateur
ADAM

Le micro-ordinateur ADAM, de la société CBS met le « traitement de texte », l'informatique et le jeu à la portée de tous.

C'est cet outil assez extraordinaire que ce livre va vous faire découvrir : c'est simple, vous ne pourrez plus, ensuite, vous en passer !

Cet ouvrage s'adresse au profane. Dans un langage clair et accessible à tous, il démystifie l'informatique et, très pédagogique, il vous enseigne comment mettre le micro-ordinateur à votre service, comment l'exploiter pour rédiger votre courrier, vos circulaires, vos notes ou vos devoirs, comment le programmer dans ce langage « Basic » qui est, aujourd'hui, le plus employé. Laissez-vous guider par ses chapitres et comme en vous jouant (en jouant aussi, d'ailleurs), vous apprendrez à maîtriser l'informatique, à travailler mieux, plus agréablement et plus efficacement, à améliorer la qualité de vos loisirs, et à vous enrichir aussi, probablement dans tous les sens du terme !





Prix : 100 F

PRATIQUE DU MICRO-ORDINATEUR ADAM

H. LILEN